WL-TR-97-3003

STORESIM: An Integrated System for Multi-Body CFD Simulations Using Unstructured, Adaptive Grids

M. P. Reddy
Computational Mechanics Company, Inc.,
7701 N. Lamar, Suite 200
Austin, TX 78752

December 1996

Final Report for Period September 1990 – August 1996

Approved for public release; distribution unlimited

FLIGHT DYNAMICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIF FORCE BASE, OHIO 45433-7562

19970613 039

DTIC QUALITY INSPECTED 3

NOTICE

WHEN GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA ARE USED FOR ANY PURPOSE OTHER THAN IN CONNECTION WITH A DEFINITE GOVERNMENT-RELATED PROCUREMENT, THE UNITED STATES GOVERNMENT INCURS NO RESPONSIBILITY OR ANY OBLIGATION WHATSOEVER. THE FACT THAT THE GOVERNMENT MAY HAVE FORMULATED OR IN ANY WAY SUPPLIED THE SAID DRAWINGS, SPECIFICATIONS, OR OTHER DATA, IS NOT TO BE REGARDED BY IMPLICATION, OR OTHERWISE IN ANY MANNER CONSTRUED, AS LICENSING THE HOLDER, OR ANY OTHER PERSON OR CORPORATION; OR AS CONVEYING ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY IN ANY WAY BE RELATED THERETO.

THIS REPORT IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

Frank C. Witzeman, Jr.
CFD Research Branch

George L Seibert.

George L. Seibert, Acting Chief
CFD Research Branch

Dennis Sedlock

Dennis Sedlock, Chief
Aeromechanics Division

IF YOUR ADDRESS HAS CHANGED, IF YOU WISH TO BE REMOVED FROM OUR MAILING LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION PLEASE NOTIFY WL/FIMC WRIGHT-PATTERSON AFB OH 45433-7913 TO HELP MAINTAIN A CURRENT MAILING LIST.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** Dec 96	**3. REPORT TYPE AND DATES COVERED** Final Report; September 90 - August 96

4. TITLE AND SUBTITLE
STORESIM: An Integrated System for Multi-Body CFD Simulations Using Unstructured, Adaptive Grids

5. FUNDING NUMBERS
C: F33615-90-C-3001
PE: 62201F
PR: 2404
TA: 10
WU: B8

6. AUTHOR(S)
M. P. Reddy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Computational Mechanics Company, Inc.
7701 N. Lamar Suite 200
Austin, TX 78752

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
FLIGHT DYNAMICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AFB OH 45433-7623
POC: FRANK WITZEMAN, WL/FIMC, 937-255-3788

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
WL-TR-97-3003

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Aerodynamic analyses of weapon separation, missile launching and crew escape problems can be accomplished with the software package STORESIM. It is based on the numerical simulation of the systems of equations which govern fluid and rigid-body dynamic motions. STORESIM represents a state-of-the-art CFD capability for simulations of multiple, moving bodies. This capability is achieved through an integrated system of modules incorporating unstructured grid generation, finite-element flow solution, six degree-of-freedom rigid-body motion, grid movement/restructuring and time-dependent results visualization. Although the overall STORESIM package is tightly integrated and presents a uniform view of the simulation to the user, the individual modules may be used as standalone programs. This report describes each module in detail, as well as several validation and demonstation cases for typical separation environments consisting of regions of compressible, viscous flow.

14. SUBJECT TERMS
Computational Fluid Dynamics, Computational Aerodynamics, CFD, Multi-Body CFD, Unstructured Grid, Finite Element, Adaptive Grid, Moving Grid

15. NUMBER OF PAGES
147

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	SAR

Contents

DTIC QUALITY INSPECTED 3

List of Figures

1 Executive Summary

This final report titled "STORESIM: An Integrated System for Multi-Body CFD Simulation Using Unstructured, Adaptive Grids" documents the effort conducted at the Computational Mechanics Company, Inc., (COMCO) under contract No. F33615-90-C-3001 with the Air Force Flight Dynamics Directorate, Wright Laboratory, OH 45433-7562. The work spans from September 1990 to August 1996. The Air Force Monitors for the project were F. Witzeman and J. Nelson. The work was conducted by COMCO in Austin, Texas. Key investigators were Dr. Stephen R. Kennon, Dr. Chittur S. Venkatasubban, Mr. Jim M. Meyering, Mr. Manas K. Deb, and Dr. Mahender P. Reddy.

The aerodynamic simulation of store separation, missile launching, and crew escape is a mission critical problem class which only recently has become amenable to computational fluid dynamics (CFD) modeling techniques coupled with high performance computing. These problems fall under the general classification of Relative Body Motion Aerodynamics (RBMA), characterized by multiple moving bodies immersed in a viscous fluid. CFD provides a unique, essential tool for analyzing these difficult problems allowing the study of arbitrary configurations without the expense of model building, wind-tunnel testing, and/or flight testing.

Numerical analysis of RBMA problems involves grid generation, grid movement/ restructuring, adaptive methods, flow simulation, high temperature gasdynamics, turbulence, rigid body dynamics, and interactive pre and post-processing. The primary goal of this effort was to couple these disciplines into a unified computational environment for analyzing dynamic separation of weapons, stages, and crew escape vehicles from supersonic and hypersonic parent vehicles. The major outcome of this research and development effort is a software package called STORESIM. The STORESIM package is a tightly integrated group of modules (TETMESH, STOREFLOW, STOREVIS) that performs grid generation, static flow simulation, six degree-of-freedom rigid body dynamics, dynamic flow analysis, body movement, grid-node movement, grid restructuring, and time-dependent results visualization.

Key accomplishments of this effort are:

1. *Mesh Generation:* A grid generator is essentially useless without a powerful technique for defining the geometry to be discretized. To this end we have developed an unstructured mesh generation module, TETMESH, that is capable of reading the geometry in a very simple network/trimmed surface format. Salient features of this module include:

    - *Triangulation:* Surface triangulation is based on the Dulanay criterion. The surface triangulator is capable of handling complex curved surfaces with cusped edges.
    - *Volume Gridding:* The volume mesher generates tetrahedral elements with an option of generating prismatic elements extruded from the surfaces. The tetrahedralization is also based on Dulanay criterion with proper face enforcement to fill the voids.
    - Zero-thickness Surface: Thin fins protruding from a missile can be approximated as single surfaces during data preparation. TETMESH automatically generates a matching surface to create a barrier and prevent fluid movement across this surface.

- Adaptivity: The meshing module is capable of refining/unrefining the mesh. Regions of maximum solution gradients are automatically refined to better capture solution in critical regions.

2. *Flow Solver:* The STORESIM package has a compressible fluid flow solver module called STOREFLOW. Key features of the STOREFLOW module are:

   - An option for ALE (Arbitrary Lagrangian Eulerian) computations. This essentially makes it possible to use moving meshes to model multiple moving bodies. This new ALE formulation is extremely robust, and can even produce accurate results for bodies launched at high speed into ambient air.
   - A very sophisticated far stream boundary condition designed to prevent spurious energy build up and consequent reflections at transonic speeds. This approach also includes modifications to account for moving mesh boundaries.
   - Euler and Navier-Stokes versions. Turbulence modeling based on a simple zero equation model has been tested for two-dimensional flows. The extension to three-dimensional turbulence modeling, although not implements should be straightforward.
   - Real gas capability, based on a 5 species (Nitrogen, Oxygen, Atomic Nitrogen, Atomic Oxygen and Nitric Oxide) model for dissociated air.
   - Multiple element types (tetrahedra, Hexahedra or Prisms). Additional element types can be easily incorporated into the code.

3. *Six-DOF Solver:* STOREFLOW is fully integrated with a six of freedom (6 dof) solver for rigid body dynamics, and a meshmover, that moves the mesh in response to body motions.

4. Node Movement: An innovative grid movement/restructuring method based on the grid-node-flow analogy and generalized edge swapping in three-dimensions. The node movement/restructuring is required when modeling body movements in the flow field.

5. *Visualization:* The visualization package, STOREVIS, is capable of post-processing structured/unstructured grids and producing color plots of slices, isosurfaces, velocity vectors, and particle traces.

STORESIM has been used to solve several complex problems. In every case, all relevant modules of the STORESIM package are used. During the development and final validation process, each module has been tested separately and as a complete package for accuracy and robustness. The final code has been transferred to Air Force Flight Dynamics Directorate. This report discusses these efforts and describes the STORESIM module in detail.

2 Introduction

The aerodynamic simulation of store separation, missile launching, and crew escape is a mission critical problem class which only recently has become amenable to computational fluid dynamics (CFD) modeling techniques coupled with high performance computing. These problems fall under the general classification of Relative Body Motion Aerodynamics (RBMA), characterized by multiple moving bodies immersed in a viscous fluid. CFD provides a unique, essential tool for analyzing these difficult problems allowing the study of arbitrary configurations without the expense of model building, wind-tunnel testing, and/or flight testing.

When multiple bodies that are moving relative to each other are immersed in a high-speed fluid stream, many complex flow phenomena exist that must be carefully modeled. In many situations, the bodies start out or become close together producing strong shocks in choking flow associated with shock/viscous interactions. These choking shocks can produce adverse aerothermodynamic affects that when coupled with the overall proximity to other bodies, can result in undesirable flight trajectories of missiles, stores, and/or crew escape vehicles. In some situations, these adverse aerodynamic effects can be severe enough to produce contact between the separating body and the parent aircraft. The study of these effects is even more important for high-speed supersonic and hypersonic aircraft of the future. These aircraft must be able to separate safely from their stores and/or missiles at high speeds.

In addition to the proximity problems associated with one or more bodies, the effects of unsteady flows are important in RBMA. Shocks, shear layers and vortices appear and disappear within a RBMA simulation. The seldom seen strong oblique shock solution of the gasdynamic equations can appear due to the motion of a body relative to the free-stream flow. To accurately model these diverse affects requires the use of modern adaptive grid technologies that are able to adapt the grid to regions of complex unsteady flow features.

Another phenomena which will encountered by future aircraft is hypersonic flow-fields with associated effects. These flows are characterized by high temperature effects that must be accurately modeled and present still another challenge to modern CFD. Finally, the effects of turbulence must be accounted for by appropriate models.

2.1 Objectives

The US Air Force has a great need for modern CFD simulation software that will quickly and accurately predict the behavior of new and/or modified weapons systems. This need has arisen due to the high cost of wind-tunnel- and flight-testing of new weapons configurations on current and future aircraft. Numerical analysis of RBMA problems involves grid generation, grid movement/ restructuring, adaptive methods, flow solution, high temperature gasdynamics, turbulence, rigid body dynamics, and interactive pre and post-processing. A successful simulation environment requires an integrated package that is easy to use, can accept standard CAD geometries from USAF databases, and is automatic and requires minimal user input or steering.

The main objective of this effort was to develop one such state-of-the-art software package to accurately and efficiently model these diverse phenomena. To this end we have developed STORESIM. As mentioned earlier, STORESIM is an integrated package of modules that performs grid generation, static flow solution, six degree-of-freedom rigid body dynamics, dynamic flow solution, body movement, grid node movement, grid restructuring, and time-dependent results visualization. Although the package is tightly integrated, and presents a uniform view of the simulation to the user, the software architecture is based on a set of modules that can be used as stand-alone programs. This report describes each of these modules in detail. In the next Section we describe the mesh generation procedure using the meshing module called TETMESH. In Sections 3 we describe the flow solver and the modifications related to implementation of the ALE scheme. In Section 4 we explain the six-dof rigid-body-dynamics solver and the mesh movement algorithm. The flow visualization package (STOREVIS) is explained in Section 5. Finally in Sections 6 and 7 we present the validation cases and demonstration cases showing vaious usages of STORESIM in typical problems encountered in real-life applications.

3 Mesh Generation

The efficient generation of *superior quality* computational grids for arbitrary configurations has long been a pacing item in CFD. The advent of unstructured tetrahedral grids has made the grid generation task an order of magnitude easier than either structured or block-structured approaches due to the inherent generality of unstructured grids. In the following we explain the capabilities of TETMESH, the STORESIM mesh generation module. Before discussing TETMESH, we digress to discuss the general three-dimensional tetrahedralization problem.

3.1 The 3D Tetrahedralization Problem

The problem can be stated as follows:

> *"Given a set of triangles whose union forms the boundary for a closed region in 3D, fill the region with tetrahedra such that the surface triangles appear as faces of the tetrahedralization."*

At first glance (and first implementation as well), one might think that this problem is trivial. However, the basic fact is that unstructured mesh generation in three dimensions is orders of magnitude more difficult than the analogous problem in two dimensions. This is due to basic theorems that state that a given polyhedron formed from triangular facets does not always have a valid tetrahedralization [19].

The basic methods for solving the above problem are *Advancing Front* and *Delaunay* (Octtree solves a different problem, since the method creates the surface triangles/nodes as part of the process). The Advancing Front method is appealing since one can practically guarantee that one can march the mesh away from the surface for at least a few layers and, therefore, *a priori* enforce that the surface triangles appear. However, one has simply pushed the problem away from the boundary into the interior by forming a (possibly more complicated) new cavity that must be filled with tetrahedra. Eventually, as borne out by practical experience with these methods, a non-tetrahedralizable cavity is often produced, and the standard method will fail since it will not be able to fill the last hole in the mesh (note: various schemes are used to try to overcome this, including backing up a few layers, etc.). The Delaunay method is no different in that it suffers from the problem right at the domain boundaries, except for the few simple cases where the surface triangulation is *Delaunay compatible* (see below) with the volume mesh. The conclusion is that for either approach, one will need (eventually) a *Face Enforcer* that can enforce the required cavity faces 100% of the time. We have implemented one such face enforcer and it is described below.

The module in STORESIM for grid generation, TETMESH, implements a constrained Delaunay procedure for surface triangulation and interior tetrahedralization. TETMESH also generates semi-structured prismatic grids by extruding surface triangles for accurate boundary-layer/viscous flow simulation.

3.2 Surface Mesh

To address the 3D tetrahedralization problem one needs to define surfaces whose union forms the boundary of the domain and triangulate these surfaces. In this subsection we describe the procedures used to define the surface and the surface triangulation method implemented in TETMESH.

3.2.1 Geometric Modeling

A grid generator is essentially useless without a powerful technique for defining the geometry to be discretized. TETMESH allows for the surface geometry to be specified in either trimmed surface or network format. The network format is a collection of quadrilateral topology of wireframes with $n_i \times m_i$ points defining their shape. Networks are joined with other networks along common edges to define the domain boundary. A network may have one or more networks joined to each of the edges, but no edge-to-edge overlap is allowed as shown in Figure 3.1. It should also be noted that the network may have degenerate edges (i.e., one of the edges can be collapsed to a point to form a triangular surface Figure 3.2).

Trimmed surfaces are defined as two pieces of information: a general surface shape description (the defining support surface), and a set of one or more boundary trimming curves which can be broken into multiple edges with common end points (Figure 3.3). The trimming curves define the actual surface shape and allows for arbitrary topology surfaces (holes, non-convexity etc.,). TETMESH converts the network format files to the trimmed surface format automatically.

3.2.2 Surface Triangulation

Surfaces to be triangulated are defined as trimmed surfaces using either wireframe support surfaces and point-defined trimming curves, or using IGES entities 126/128 and 141/143. These trimmed surfaces are discretized in two steps: a) first the user defines the spacing on each edge (trimming curve) of each surface using several available options (uniform, input, cluster) b) secondly, the user defines the required spacing control of triangles on the surface (flatness measure, curvature, spacing), c) finally, the user requests surface triangulation. Surfaces can be re-triangulated if the spacing or quality is not acceptable.

Surface triangles are generated using a 2-1/2 dimensional Delaunay scheme as follows:

1. Triangulate all the boundary nodes in parameter space

2. Transform to physical space

3. Swap diagonals to ensure that the *Surface Delaunay Property* (SDP) is satisfied (see below)

4. Incrementally insert new nodes into the triangles with the largest minimal circumspheres in turn as follows:

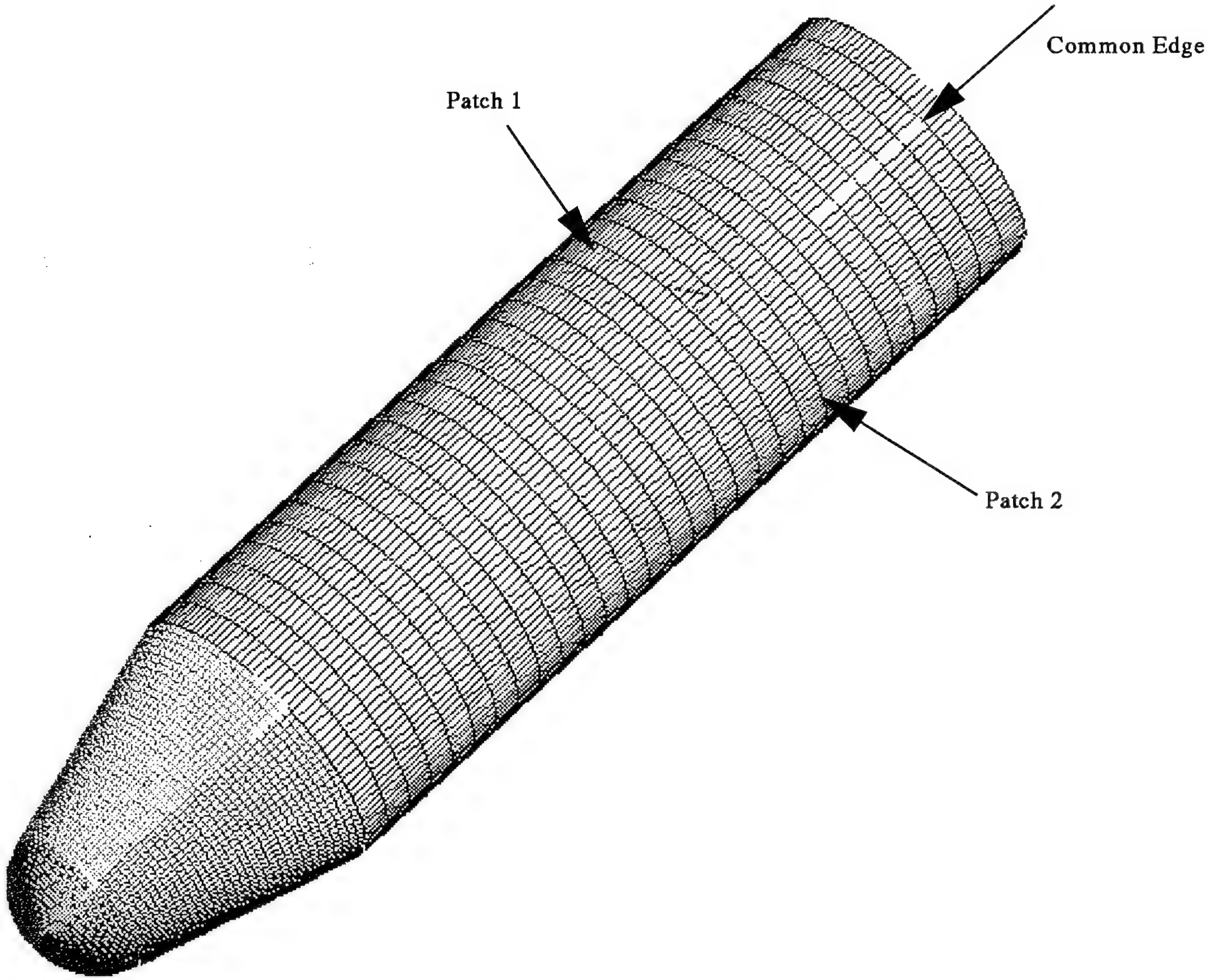


Figure 3.1: The network format of surface patches. In this figure we show the surface definition of a forebody an assembly of network patches.

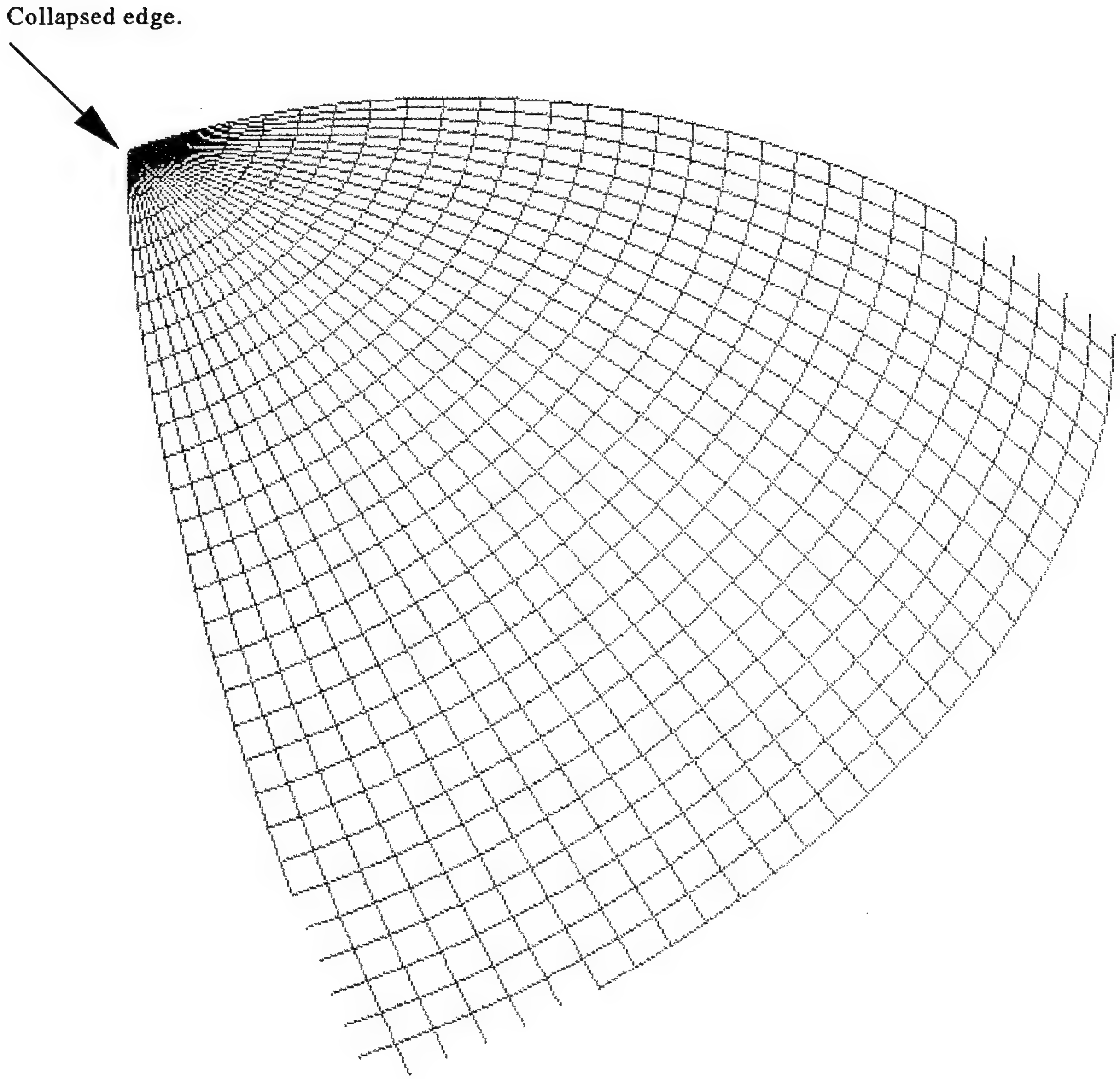
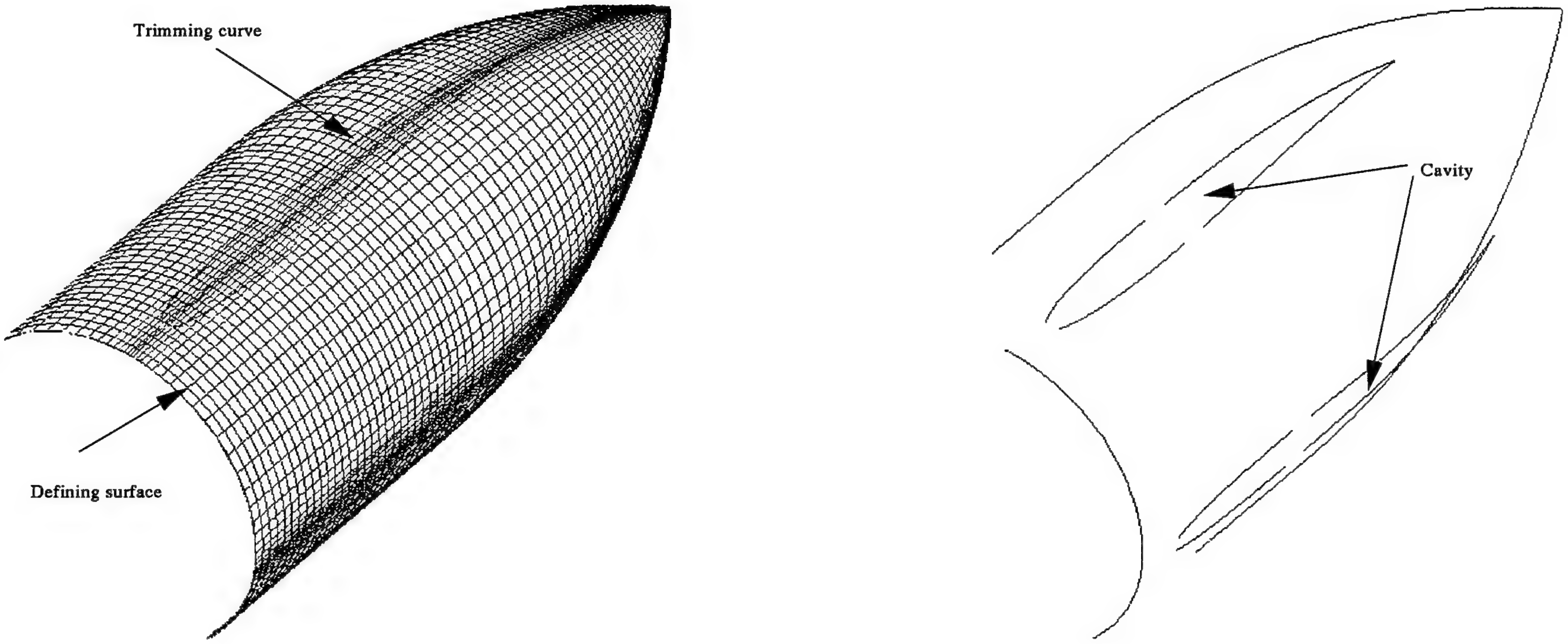


Figure 3.2: A network patch with a degenerate (collapsed edge). TETMESH allows use of such patches to define the surface boundary.



a: Original network format of the surface.

b: Surface generated after removing parts of the surface with trimming curve.

Figure 3.3: Trimmed surface generated using trimming curves.

- The new node has (x, y, z) coordinates of the triangle's circumsphere
- Find a triangle in parameter space containing the new node
- Create three new triangles from the new node and the containing triangle
- Swap diagonals to regain the SDP

The process is continued until the desired triangle spacing is achieved. This procedure has a great advantage over simply triangulating in parameter space. Parameter space triangulations suffer from the distortions of mapping to physical space. Thus, if one tries to achieve nice triangulations in parameter space, there is no guarantee that the triangulation will be acceptable in physical space. Our procedure produces Delaunay surface triangulations, and thus achieves the same quality benefits as planar Delaunay surface triangulation [16].

3.2.3 Sub-Dimensional Delaunay Property

The Delaunay criterion for a simplicial mesh is that no node lies in the strict interior of any simplex circum-hypersphere. This concept is applicable to any space dimension. The circum-hypersphere is defined as the hypersphere in R^n that passes (uniquely) through the $n + 1$ vertices of the n-simplex. The hypersphere is unique up to degeneracies that occur due to pathological placement of nodes (e.g., all nodes in an m-dimensional hyperplane, $m < n$). One can define a locus of sub-dimensional hyperspheres for any dimension $m < n$, which we term the set of m-sphere's for a given n-simplex. For example, a line segment (1-simplex) embeded in 2D has a set of 1-spheres that have their centers along a line that passes through the mid-point of the line segment (the center of the 1-sphere in the line-segment's coordinate space) and perpendicular to the segment.

Theorem 1: An m-simplex will appear in an n-dimensional Delaunay triangulation if and only if there exists an m-sphere of the m-simplex that does not contain any vertices of the triangulation.

We can now state the:

> Surface Delaunay Property (SDP): a surface triangulation is Delaunay if its triangles' minimal circumspheres do not contain other nodes.

The minimal circumsphere of a triangle embedded in 3D is just the sphere passing through the three nodes with center at the center of the circumcircle of the triangle lying in the plane of the triangle (alternatively, we see that it is the minimal radius sphere passing through the three triangle vertices).

The SDP gives a concrete criterion for determining which surface triangulations are *compatible* with a 3D tetrahedral Delaunay grid computed from the surface nodes. It is not surprising that most of the difficulty that many researchers have had with 3D constrained Delaunay meshing is due to the fact that most surface triangulations are simply incompatible with the volume mesh.

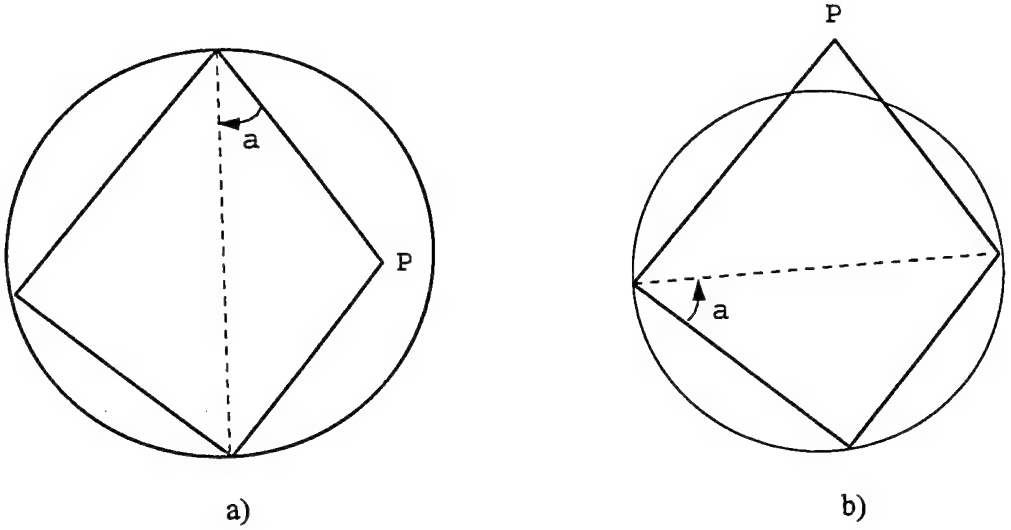


Figure 3.4: Edge swapping criterion for a pair of triangles. a) deos not satisfy the Delaunay criterion b) satisfies Delaunay criterion.

Note that we use the minimal circumsphere as the criterion for determining the Delaunay property of a surface triangulation. This is only a sufficient condition, however, and simply states that if the minimal condition is satisfied, the mesh will be Delaunay, and the surface triangles will appear in the volume mesh.

3.2.4 Diagonal Swapping

We achieve a Delaunay surface mesh (from a non-Delaunay one) by swapping diagonals on the surface for each pair of triangles sharing an edge that contain the opposite node in the triangle's circumsphere center. Edge swapping consists of modifying an existing triangulation by re-connecting nodes. In two-dimensions, pairs of triangles are examined and the resulting quadrilateral's diagonals are swapped if the result produces a Dulanay mesh (see Figure 3.4).

3.2.5 Extrusion

Once the surface triangles have been generated, the surface can be extruded by user specified criteria to produce a layer of prismatic elements on the surface boundary. This layer consists of n_l sub-layers of prisms that produce a semi-structured mesh suitable for accurate viscous flow computations in the near-body region. Clustering functions can be applied to achieve a desired first-layer spacing off the body.

Nodes on the extruded components are generated in a marching scheme away from the body. The scheme is given as follows:

1) Compute the surface normals at each node.

2) Smooth the surface normals by averaging with each neighboring node (from the connectivity graph of the surface triangulation)

   – smoothing is user controlled by the number of passes and a relaxation factor

3) Create each layer away from the body from the previous one:

   a) new point location is given by $x_{i+1} = x_i + d_i * n_i$ where i is the layer number, d_i is the layer spacing and n_i is the normal for layer i at the given node

   b) After computing the new node location, the new normal is computed by smoothing it from its neighbors as was done for the initial surface triangulation normals.

4) after nodes have been created, create the prisms in the layer

This scheme is augmented to allow for the addition of new nodes due to face enforcement or adaptivity (see below). Figure 3.5 shows prismatic elements extruded from a triangulated surface of a sphere.

3.2.6 Zero-thickness Surfaces

When modeling fluid flow problems, we commonly encounter thin mounted surfaces such as fins on a missile. In certain instances the top and bottom surfaces of such fins may be so close to each other that it would be beneficial (without loss of flow solution accuracy) from the user perspective to represent these as a single surface. In TETMESH we call such surfaces a `zero-thickness` surfaces and represent these as a single trimmed surface. For each zero-thickness surface, after triangulation, TETMESH creates an new surface with identical surface nodes and triangle distributions such that the normals are pointed in opposite direction. These newly created surfaces are called *buddy surfaces.* These two collapsed surfaces form the top and bottom boundaries of the zero-thickness surface and prevent any fluid interaction across the boundary. An example of the zero-thickness surface in shown in Figure 3.6a. In this example the flap (called *Thin_Fin*) coming out of the cube is modeled as a zero-thickness surface. Figure 3.6b shows the surface normals for the original surfaces and the newly created buddy surface. Figure 3.6c shows the surface triangulation and the surface normals for the nodes on the original surface and the new surface created by TETMESH.

3.3 Volume Mesh

The volume mesh is computed from the positions of the (possibly extruded) surface nodes and triangles. We first tetrahedralize the space between the inner and outer surfaces using the scheme described below. This forms a tetrahedralization of the boundary nodes which then can be checked to see if any surface triangles need to be enforced. The enforcement of any missing triangles is performed as discussed below. Once all surface triangles exist in the tetrahedralization, interior nodes are added at tetrahedra circumsphere centers as was done

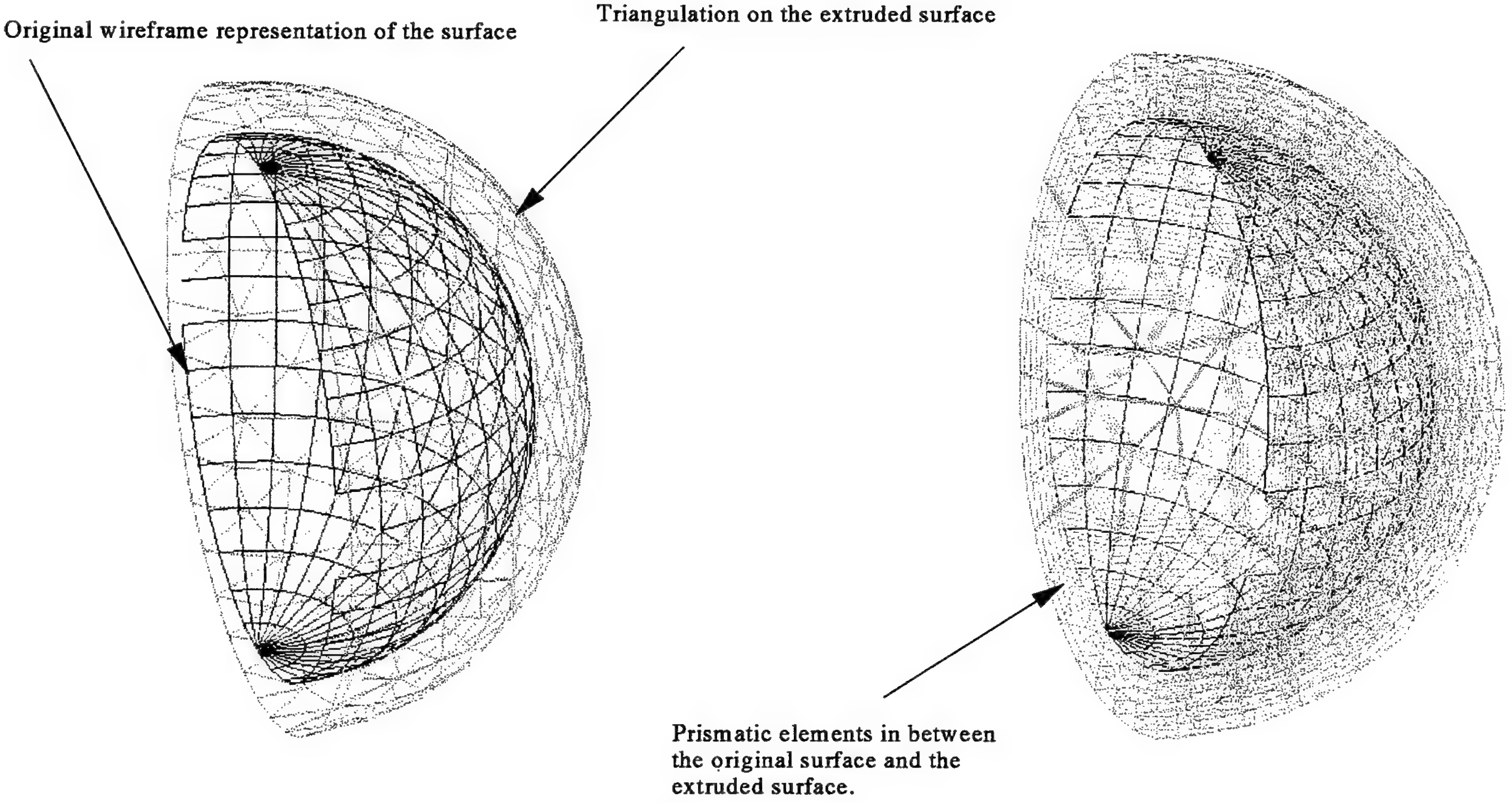
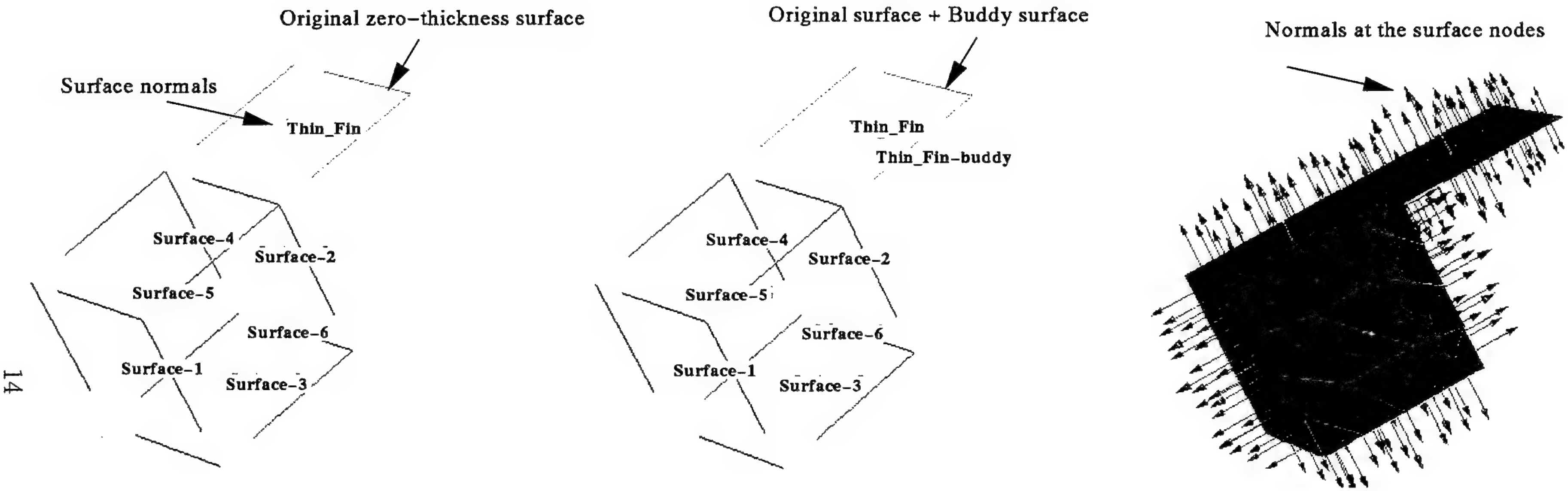

Original wireframe representation of the surface
Triangulation on the extruded surface
Prismatic elements in between the original surface and the extruded surface.

Figure 3.5: Illustration of prismatic elements created by extruding the surface triangles normal to the surface.



6(a)

6(b)

6(c)

Figure 6: Illustration of zero−thickness surface modeling capability. a) Initial surface patches before triangulation. b) After ttrahedralization: Notice that an identical surface (Thin_Fin−buddy) is created. c) Normals (into the fluid domain) at the nodes on the surface.

for the surface triangulation. We add nodes in the tets that have the largest circumsphere radius until the desired spacing is achieved. Currently, tetrahedral spacing is controlled by a linear interpolation of the surface triangle spacing into the interior (defined by the boundary tetrahedralization), although it would be quite easy to incorporate a user-supplied spacing function. However, we have found that with adaptivity, it is only necessary to achieve a reasonably good mesh to start with, and the automatic method for spacing control in TETMESH suffices for that purpose.

3.3.1 Volume Tetrahedralization

We use a modified version of point insertion followed by diagonal swapping extended to 3D. The method is modified from the original as described in [2, 15]. The basic idea is to find the tetrahedron that contains the new node in its strict interior, form four new tetrahedra in the interior of that tet (by joining the new node to the four faces of the tet), then performing *three-dimensional edge swapping* (3DES). 3DES is an implementation of Lawson's observation [17] that there are only 2 unique triangulations of any configuration of $n+2$ points in n−D space. Thus, in 3D, two tets sharing a face consist of five vertices, and the opposite vertices can be connected to form an alternate tetrahedralization of the five nodes. Moreover, three tets sharing an edge can (sometimes) be swapped back to form two tets sharing a common face. Lawson also showed that there is only one unique triangulation that satisfies the Delaunay property (modulo degeneracies). The implementation of this edge swapping algorithm requires the following:

1) $2 \longrightarrow 3$ swap: swap two tets sharing a face into three tets sharing a common edge; this can only be performed if the two tets form a convex polyhedron

2) $3 \longrightarrow 2$ swap: swap three tets sharing an edge to form two tets sharing a face; this can only be performed if the resulting polyhedron is convex

3) $4 \longrightarrow 4$ swap: four tets sharing a common edge are swapped to form four new tets; this case must be implemented to deal with the case where the polyhedron of cases 1) and 2) is degenerately convex.

4) $2 \longrightarrow 2$ swap: two tets on a boundary sharing a common edge are swapped to give two new tets.

3.3.2 Surface Classification

We can identify which triangles will not appear in the Delaunay mesh using Theorem 1 above as follows. First, we classify all triangles whose minimal circumspheres don't contain other nodes as type A. These triangles will appear automatically as faces of the volume mesh. The rest of the triangles are of type B. Now, we further classify the type-B triangles as follows:

- If there exists a 2-sphere for the triangle that does not contain another node, classify it as B.1, else classify it as B.2

- Furthermore, for B.2 triangles, we classify them as:

B.2.1: the forming points of the triangle are not violators of any other surface triangle.

B.2.2: one or more forming point of the triangle is a violator of another surface triangle.

Using this classification, we insert each node of the surface triangle-by-triangle until they are all inserted as follows:

1) Insert all type-B.2.2 triangles; insert triangle edges and faces into the face checker's hash table.

2) Insert all other type B triangles; insert edges and faces into the face checker hash table.

3) Insert all non-inserted nodes with the face checker ON.

The face checker uses a map between triangle edges (pairs of vertices) and tet faces (triples of vertices) to ensure that once an edge or face of a required surface triangle is in the tetrahedralization, it will not be removed by any subsequent addition.

Note that he B.2.2 surface triangles are the most difficult to enforce since their nodes potentially violate the Delaunay property of other surface triangles.

The tetrahedralization process can be modeled by simulating the addition of the nodes in the order given above. This simulation then gives the user an estimate *before the tetrahedralization* of how many unenforced faces will exist. Additional research is needed on further utilizing the triangle classification for finding an optimal insertion order within a particular sub-type. We speculate that an ordering can be found for triangle insertion that will further minimize the number of unenforced faces that have to be enforced.

3.3.3 Face Enforcement

The above procedure for triangle classification and ordered insertion can still lead to situations where the volume mesh does not contain some of the required surface triangles. We enforce these remaining triangles in a two step process.

As noted by Hazlewood [9, 10, 11] (and later by Weatherill [24]), one can enforce the triangles by finding the intersection of the triangles edges and face with the existing mesh, then inserting new nodes at the locations of the intersections. Here's the basic algorithm:

1) Enforce all three edges of the required surface triangle. If triangle edge E does not already exist (possibly with additional vertices along it) then that edge pierces at least one tetrahedron. Subdivide each pierced tetrahedron to create the segment of edge E passing through it.

2) Enforce the required triangle face. If face ABC does not already exist (possibly containing vertices on its boundary that were created in step 1) then that face intersects at least one tetrahedron. Subdivide each intersected tetrahedron to create the portion of face ABC that intersected it. Note that the portion of ABC that is common to an intersecting tetrahedron

may be triangular or quadrilateral. The union of these portions of triangle ABC then form the required face to be enforced.

In the above, each intersected tetrahedron is subdivided into smaller tetrahedra with the constraint that the specified edge or face be a part of the resulting sub-tetrahedralization. When there is more than one possible sub-tetrahedralization, a Delaunay one is chosen.

The next step is to remove newly added nodes by collapsing edges between such nodes and other nodes in the enforced face.

1) for each new face (interior) node, try to collapse it to another face node, another edge interior node, or a node of the original face.

2) for each edge interior node, collapse the node to one of its neighbors on the edge.

The collapsing procedure is described below. The reason that we try to remove the new nodes added by the face enforcer is that they can be arbitrarily close to other nodes, and thus produce arbitrarily small (and thus bad) triangles and tets.

3.3.4 Edge Collapsing Procedure

A major contribution of this project has been the development of the following algorithm for collapsing edges to remove unwanted nodes. Other schemes for node removal require re-tetrahedralizing the cavity that results from node removal [24]. It is well known that arbitrary cavities cannot be tetrahedralized while maintaining the boundaries. This is just another example of the difficulty of 3D tetrahedral mesh generation. Moreover, merely determining whether a configuration is tetrahedralizable has been shown to be NP-Complete [19]. However, the following constructive algorithm provides both a test for tetrahedralizability as well as the resulting tetrahedralization, *in the case that one starts with an existing tetrahedralization, and removes a given node.*

1) for each possible collapse, check if the resulting tetrahedralization is valid: each tet that doesn't have a collapsed edge must have positive volume 2) continue until a valid collapsed configuration is found 3) remove all tets that have a collapsed edge, and also the newly inserted node.

This edge-collapsing procedure is also used in STORESIM's adaptive unrefinement strategy (see below). The above procedure could be modified to not only choose a valid configuration, but to choose the best quality tetrahedralization.

3.3.5 Boundary Tet Layer

As is done in the Rampant code [3], we introduce a layer of nodes right at the boundary before face enforcement to produce an optimal set of tets near the boundary. The nodes are placed along a normal to each face circumcenter at a distance of $r\sqrt{2}$ where r is the minimal circumsphere radius. This initial layer produces good quality tets right at the boundary and furthermore reduces the possibility of badly shaped triangles/tets being generated by the face enforcer. No edge or face of an existing surface triangle is removed by this process since the 'face checker' is on during the node addition.

3.3.6 Mesh Quality

We define the aspect ratio of a tetrahedron by the radius of its inscribing sphere divided by the radius of the circumscribing sphere. This measure is scaled so that an equilateral tetrahedron has aspect ratio 1, and a zero-volume, degenerate tet – or sliver – has aspect ratio 0.

Slivers are removed by a process in which an edge of the degenerate tet is deleted. This edge removal is precisely the $3 \longrightarrow 2$ swap in its simplest form, but in general it is an $n \longrightarrow 2n-4$ swap. It is described as follows. For each edge of the tet we're trying to remove, consider all tetrahedra that share that edge. Suppose the candidate edge is surrounded by n tets. Those tets are defined by $n+2$ nodes, two of which define the shared edge. Enumerate all possible triangulations of the n nodes not on the shared edge. The triangulation takes into account *no* geometry other than the cyclic ordering of the nodes. That is, for the purpose of enumerating the triangulations, one could assume that the n nodes lay on a circle. For each of those triangulations (consisting of $p = n - 2$ triangles), we form the resulting $2p$ tetrahedra by joining each triangle with the two nodes of the shared edge. This obviously removes the edge. Of the possible resulting tetrahedralizations, we choose the one that maximizes the minimum aspect ratio of the $2p$ new tetrahedra. It should be mentioned that although this procedure is usually effective, it's not always possible to remove an edge at all, and even if it is possible, doing so may produce *worse* tets. Also, due to the exponential complexity of enumerating all triangulations, this procedure is effective only for very small n, say $n < 10$.

3.4 Adaptivity

For accurate simulation of both steady and unsteady flows, one must use an adaptive meshing to capture important flow features such as shock waves and boundary layers. A critical component of an adaptive code is the ability both to refine and to unrefine the mesh as outlined below.

3.4.1 Refinement

Adaptive refinement of the mesh is accomplished by adding new nodes on selected edges of the mesh. This creates $2n$ new tets where n is the number of tets surrounding the broken edge. The edges are selected for refinement based on typical gradient error indicators (such as the density gradient).

3.4.2 Unrefinement

Unrefinement is implemented in STORESIM by identifying nodes to be removed. This is accomplished by marking all nodes as `NOT-REMOVED` then marking which edges have error indicators smaller than the `UNREFINE-CUTOFF` value. Each edge then marks its two nodes, and if all edges surrounding a node should be unrefined, then the node is marked for unrefinement. Next, the node is collapsed to one of its neighbors by selecting the neighbor that produces the best resulting mesh. Note that this procedure requires *no* cavity-filling as do other schemes [8].

Also, no expensive tree structure of tets needs to be implemented, since whenever the mesh is refined, the old tets are simply discarded. The old tets are easily recovered by the edge collapsing procedure (unrefinement) as described above.

3.4.3 Surfaces

When a new node is added on an edge during refinement, it is possible that the edge lies on the surface boundary. Thus, we must update the surface triangulation (creating four new surface triangles from the two sharing the edge) as well as ensure that the new node lies on the actual surface and not simply on the straight line between the two edge endpoints. This ensures high surface fidelity.

Surfaces that have extruded prisms must have the prisms adapted as well. We produce 4 new prisms for each original 2 in each layer of the extrusion, and ensure that the node on the interior boundary lies on the actual surface.

3.5 Interface to Other Codes

Once the mesh generation is completed, TETMESH allows the user to save the mesh in several different forms. This enables the user to output the mesh and use it with a third-party flow solver. At the present time, TETMESH is capable of outputting the mesh in the following formats:

i) Internal binary/ascii format to be read into TETMESH,

ii) Gamma format: to input data to stand-alone STOREFLOW solver,

iii) FAST format,

iii) COBALT format, and

iv) ascii format listing tets and their faces.

Of these, the FAST format is widely published.

4 Flow Solver

4.1 Background

STOREFLOW was developed within the context of the STORESIM project, whose primary goal was the numerical computation of the trajectories of stores (i.e., missiles, bombs, etc.,) released from flying aircraft. Any solver employed for such a purpose must meet several stringent requirements. Firstly, it should be capable of modeling very complex and arbitrary geometries that characterize "stores" and aircraft. It must compute physically relevant flow solutions over these configurations over a wide range of speed regimes, from low subsonic to transonic, supersonic and hypersonic Mach numbers. Finally, it must deliver solutions for multiple bodies, moving at varying relative speeds. This last capability differentiates it from traditional flow solvers that are mainly used to compute flows at some fixed Mach number.

Keeping the above requirements in mind, the algorithm used to develop STOREFLOW was based on the SUPG (Streamline Upwind Petrov Galerkin) method [14] , [22], which has in recent years emerged as a very robust approach for Finite Element Computational Fluid Dynamics. STOREFLOW contains several new features to enhance its capabilities. Among the most important in this list are:

- A modification [23] to allow ALE (Arbitrary Lagrangian Eulerian) computations. This essentially makes it possible to use moving meshes to model multiple moving bodies. This new ALE formulation is extremely robust, and can even produce accurate results for bodies launched at high speed into ambient air.

- A very sophisticated far stream boundary condition designed to prevent spurious energy build up and consequent reflections at transonic speeds. This approach also includes modifications to account for moving mesh boundaries.

- Euler and Navier-Stokes versions. Currently, the viscous, Navier-Stokes code in operational only for laminar flow. Turbulence modeling based on a simple zero equation model has been tested for two-dimensional flows. The extension to three-dimensional turbulence modeling, although straightforward, is not yet operational.

- Real gas capability [5], based on a 5 species (Nitrogen, Oxygen, Atomic Nitrogen, Atomic Oxygen and Nitric Oxide) model for dissociated air.

- Fully integrated with a 6 degree of freedom (6 dof) solver for rigid body dynamics, and a mesh-mover, that moves the mesh in response to body motions.

- Multiple element types (tetrahedra, Hexahedra or Prisms). Additional element types can be easily incorporated into the code.

4.0

4.2 Governing Equations

The equations governing the flow of a compressible, viscous fluid are the Navier-Stokes equations and are given by the vector relation:

$$\boldsymbol{U}_{,t} + F_{i,i} = F^d_{i,i} \tag{4.1}$$

where

$$\boldsymbol{U} \text{ : vector of conservation} = \rho \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e \end{Bmatrix}$$

ρ: fluid density

u_j: j =1,2,3 = fluid velocity components

e: total energy per unit mass = $C_v\theta + \frac{1}{2}u_iu_i$

θ: temperature of the fluid

C_v: specific heat at constant volume

F_i: Euler flux vector and is given by:

$$F_i \ = \ u_i\boldsymbol{U} + p \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{Bmatrix} \tag{4.2}$$

F^d_i : diffusive flux vector and is given by:

$$F^d_i = \begin{Bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij}u_j - q_i \end{Bmatrix} \tag{4.3}$$

τ_{ij} : is the stress tensor = $\lambda u_{k,k}\delta_{ij} + \mu\left(u_{i,j} + u_{j,i}\right)$

μ : viscosity coefficient

λ: bulk viscosity coefficient = $-\frac{2}{3}\mu$

q_i : heat flux component =$-k\theta_{,i}$

p: pressure

4.2.1 Euler Equations

Consider an inviscid, adiabatic flow field. The equations governing the flow of such fluid can be obtained from the Navier-Stokes equations by neglecting all shear stress terms and heat conduction terms. The resulting set of equations are called Euler equations. Thus, the Euler equations are:

$$\boldsymbol{U}_{,t} + F_{i,i} = 0 \tag{4.4}$$

The Euler equations are used for flows at high Reynolds numbers. From Prandtl's boundary layer analysis, this is a valid approximation for such flows outside the viscous regions close to the surfaces.

4.2.2 Navier - Stokes Equations in an ALE (Arbitrary Lagrangian Eulerian) Reference Frame

In this section we present the Navier-Stokes equations in an ALE reference frame. The Navier-Stokes equations as presented in (4.1) are only valid in a fixed frame of reference. This is the conventional "Eulerian" framework. Thus, the flow solution is defined at arbitrary points "p", that are stationary with respect to an inertial frame of reference [x,y,z] (see Figure 4.1).

In the "Lagrangian" representation, the point "p" has a velocity u_i^g, that is exactly the same as the fluid velocity u_i. However, in an ALE approach, u_i^g is completely arbitrary. It is advantageous to use such an ALE representation to model the flow over multiple bodies that are moving at arbitrarily different speeds. The Navier-Stokes Equations for the ALE frame can be shown to be:

$$\boldsymbol{U}_{,t} + F_{i,i} = -u_{i,i}^g \boldsymbol{U} + F_{i,i}^d \tag{4.5}$$

The first term on the R.H.S of (4.5) is a grid dilation term. The flux vector F_i, is defined as:

$$F_i = F_i^e - u_i^g \boldsymbol{U} \tag{4.6}$$

where F_i^e has the same form as the flux vector in an Eulerian frame. It has been shown [23] , that the grid dilation term may be conveniently eliminated to give the ALE equation in the form:

$$\boldsymbol{U}_{,t} + [A_i - u_i^g I]\, \boldsymbol{U}_{,i} = F_{i,i}^d \tag{4.7}$$

where

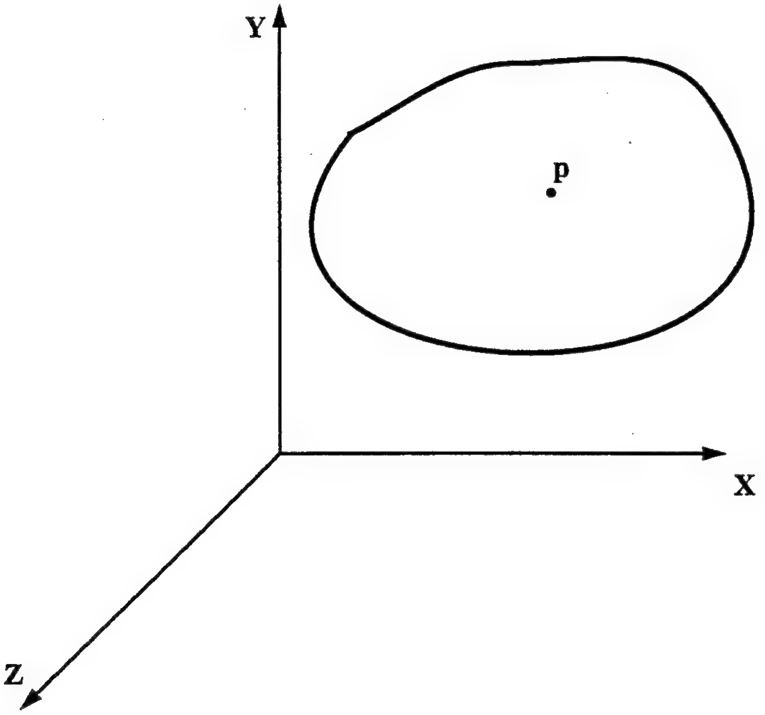


Figure 4.1: Inertial Reference Frame

$$A_i = \boldsymbol{F}^e_{i,u} = Euler\ Flux\ Jacobian \tag{4.8}$$

Equation (4.7) forms the basis of the STOREFLOW code.

4.3 Boundary Conditions

Symmetry and specified heat flux conditions are imposed as natural boundary conditions, by merely computing the associated boundary integrals. The "no-slip" condition on solid walls is imposed as the "Dirichlet" conditions, $u_i = u_i^g$. At farstream boundaries the formulation applies specified farstream conditions in conjunction with a non-reflective, characteristic theory [21].

4.4 Weak Formulation of the ALE Navier-Stokes and Euler Equations, in the Context of the SUPG Method

The weak form of these equations , in the starting point for a Finite Element discretization and solution algorithm. The intent here is to present the weak form in sufficient detail to facilitate an in depth examination of the corresponding computer code. Precise details on steps required to arrive at this formulation may be found in Reference [14],[22], and [23].

For the ALE Navier-Stokes equations we employ the weak form:

$$\begin{aligned}\int_{\Omega} W \cdot (\boldsymbol{U},t + [A_i - u_i^g I]\, u,_i)\, d\Omega + \int_{\Omega} W,_i \cdot F_i^d \cdot d\Omega - \int_{\Gamma_\infty} W \cdot (F_i - F_{i\infty})\, n_i d\Gamma \\ - \int_{\Gamma_{Wall}} W \cdot (F_i - F_{ip})\, n_i d\Gamma - \int_{\Gamma_{Wall}} W \cdot F_i d_{ni} \cdot d\Gamma + I_{SUPG} + I_{shock} &= 0\end{aligned}$$

where

$$F_{ip} = P \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i^g \end{Bmatrix} \tag{4.9}$$

- W = vector of weighting functions chosen from a space of compact support
- I_{SUPG} = Streamline Operator
- I_{shock} = Shock Operator

The streamline and shock operators are defined in Appendix-A.

4.5 Finite Element Discretization

A Finite Element discretization based on linear elements has been implemented for various element types. The elements considered are triangle and quadrilateral in two-dimensions, and tetrahedra and prisms in three-dimensions. A semi-discrete formulation leads to the following representation for the time derivative of the solution variable at each node, at time $t = t_n$:

$$[M^n] \left\{ \dot{\bar{u}}^n \right\} = -R^n \tag{4.10}$$

where M^n is a mass matrix assembled as follows:

$$M^n = \mathbf{A}_{\mathbf{e=1}}^{\mathbf{Nel}} \left[\mathbf{M^e} \right] \tag{4.11}$$

where Nel = number of elements,

$$M^e = \left[M_{pq}^e \right], p, q = 1, 2, \ldots, N_{en} \tag{4.12}$$

where N_{en} = number of elements nodes,

$$M_{pq}^{e} = \int N_p^e N_q^e [I] d\Omega \tag{4.13}$$

where I is the 5×5 unit matrix and N_p^e, N_q^e are linear element shape functions. Also, $\bar{u} = 5 \times N_{np}$ vector of conservation variable at all nodes.

Finally, R^n is a $5 \times N_{np}$ residual vector, which we assemble as follows:

$$R^n = A_{e=1}^{Nd} \left\{ R^e \right\} \tag{4.14}$$

$$R^e = \left[R_p^e \right], p = 1, 2, ..., N_{en} \tag{4.15}$$

$$\begin{aligned} R_p^e = \int_{\Omega^e} N_p^e \cdot \left[\tilde{A}_o V_{,t} + \tilde{A}_i^{ALE} V_{,i} \right] d\Omega + \int_{\Omega^e} \left(I_{SUPG} + I_{shock} \right) d\Omega \\ - \int_{\Gamma^e_{infty}} N_p^e \cdot \left(F_i - F_{i\infty} \right) n_i d\Gamma - \int_{\Gamma_p^e} N_p^e \left(F_i - F_{ip} \right) n_i d\Gamma - \int_{\Gamma_p^e} N_p^e \cdot F_i^d \cdot n_i d\Gamma \end{aligned} \tag{4.16}$$

where

$$\left. \begin{array}{l} V \equiv vector\ of\ entropy\ variables \\ \tilde{A}_o, \tilde{A}_i^{ALE} \equiv flux\ matrices \end{array} \right\} Defined\ in\ the\ Appendix - A$$

4.5.1 Time Discretization

The semi-discrete form (4.10) has been used to developed an explicit Runge-Kutta and implicit schemes of varying orders of accuracy. For example, the general "α" scheme for computing the solution at the $(n+1)^{th}$ time step is:

$$u^{n+1} = u^n + \Delta t \left(\alpha\, \dot{u}^{n+1} + F_\alpha\, \dot{u}^n \right) \qquad 0 \leq \alpha \leq 1 \tag{4.17}$$

if

$\alpha = 0 :$ *Explicit, Forward Euler*
$\alpha = 1 :$ *Implicit, Backward Euler*
$\alpha = \frac{1}{2} :$ *2^{nd}order accurate, Cranck – Nicholson scheme*
In a similar manner, Runge-Kutta schemes of varying orders ($3 \sim 5$) have been coded.

4.5.2 Non-Linear GMRES, IMPLEX-Solver

An Implex (Implicit/Explicit) solver, for the Cranck-Nicholson scheme has been developed. This automatically picks Implicit and Explicit nodes in the mesh based upon a user supplied range for the CFL coordinates as follows:

- Choose CFL_{min} and CFL_{max}
- Compute $\Delta t_e = \frac{CFL_{max} \cdot h_e}{|Vta|_e}$ for each element "e" , where
  $h_e = element\ length\ parameter$
  $V = velocity$
  $a = Speed\ of\ Sound$
- Compute Δt_i for each node as the minimum value of the elements connected to that node. Then find $\Delta t_{min} = min(\Delta t_1, ..., \Delta t_i)$, which is the timestep used to advance the solution.
- For each node compute $CFL_i = \frac{CFL_{max}}{CFL_{min}} \cdot \frac{\Delta t_{min}}{\Delta t_i}$

-
  ```
  If $CFL_i$ is less than $CFL_{min}$
      Node "i" is explicit
  else
      Node "i" is implicit
  endif
  ```

Both the fully implicit Backward Euler and the implicit algorithm require the solution of a large set of Non-linear equations, which can be represented as:

$$\tilde{N}(\tilde{u}) = \{0\} \tag{4.18}$$

This equations has $5 \cdot N_{np}$ unknowns.

A Newton-Raphson linearization yields

$$\begin{aligned} \tilde{N}\left(\tilde{u}^k + \Delta\tilde{u}^k\right) \cong \tilde{N}\left(\tilde{u}^k\right) + \left[\frac{\partial \tilde{N}}{\partial \tilde{u}}\right]^k \Delta\tilde{u}^k \approx 0 \\ \tilde{u}^{k+1} = \tilde{u}^k + \Delta\tilde{u}^k \end{aligned} \tag{4.19}$$

One needs to solve this arge linear system of equations, for the Kth approximation to $\tilde{u}$.

$$\left[\frac{\partial \tilde{N}}{\partial \tilde{u}}\right]^k \Delta\tilde{u}^k = -\tilde{N}\left(\tilde{u}^k\right) \tag{4.20}$$

This is repeated until $\Delta\tilde{u}^k$ becomes very small, for some value k. In practice, (4.20) is very expensive in terms of computer memory. Thus, recourse is available through the Nonlinear GMRES (Generalized Minimum Residual) Method [4] & [25]. In this method, the GMRES algorithm is used to compute $\Delta\tilde{u}^k$ in (4.20), without actually storing the large matrix $\left[\frac{\partial\tilde{N}}{\partial\tilde{u}}\right]^k$. Basically, in the GMRES procedure, one needs to take the product $\left[\frac{\partial\tilde{N}}{\partial\tilde{u}}\right]^k \cdot p$, where p is a Krylov vector (see references above). This product is approximated to second order accuracy in the Nonlinear GMRES procedure as follows.

$$\left[\frac{\partial\tilde{N}}{\partial\tilde{u}}\right]^k \cdot p \cong \frac{\tilde{N}\left(\tilde{u}+\epsilon p\right) - N\left(\tilde{u}-\epsilon p\right)}{2\epsilon} \tag{4.21}$$

4.5.3 Preconditioning

A preconditioner for (4.20) based on an approximate inverse to $[\frac{\partial\tilde{N}}{\partial\tilde{u}}]^k$ has been developed. The simplest often is to assume a preconditioner $[K]^{-1}$, such that $K = BlockDiag\left[\frac{\partial\tilde{N}}{\partial\tilde{u}}\right]^k$. This makes the inversion very fast. Another preconditioner that has been implemented is [18].

$$K = \left[\frac{\partial\tilde{N}}{\partial\tilde{u}}\right]^k \tag{4.22}$$

where $K^{-1} = L \cdot n$ and K^{-1} is computed such that the "fill -in" during factorization is ignored.

4.6 Flow Solver Capabilities

As mentioned earlier, STORESIM contains a module STOREFLOW that implements a vectorized version of the Streamline Upwind Petrov-Galerkin (SUPG) method [13]. The SUPG method has been modified (as outlined in [23]) to incorporate the extra terms that arise from the mesh movement in an Arbitrary Lagrangian-Eulerian (ALE) formulation of the weak form of the Navier-Stokes equations. Key features of STOREFLOW are:

- Physics:
  - inviscid (Euler)
  - viscous (Navier-Stokes)
    - laminar
    - turbulent - 0-equation model
  - real gas
    - equilibrium chemistry

        - air model with 5 species

- Time:
    - steady-state
    - time-accurate
- Solver:
    - explicit
    - implicit
    - implicit/explicit (IMPLEX)
- Mesh:
    - static mesh
    - dynamic mesh
        - ALE formulation

Any combination of the above physics, time stepping, solver or mesh can be chosen.

STOREFLOW has an element library that includes a linear triangle, quadrilateral, prism, tetrahedron, hexahedron elements. It has a unique software architecture in which a single source code is maintained which is pre-processed (by the Unix m4 macro processor) to produce actual FORTRAN code for 2D or 3D, any element type (or combination of types such as tets and prisms), or time-stepping scheme (explicit, implicit, IMPLEX) as desired.

5 Six-DOF Rigid Body Dynamics Solver

STORESIM has a fully integrated (coupled) six degree-of-freedom rigid body dynamics (6DOF-RBD) solver that uses the consistently computed forces and moments on the specified moving bodies to accurately predict the trajectory of the moving bodies under either free or powered flight. The mass properties of each moving body, thrust and moment forces as well as optionally prescribed (captive) trajectory are specified in a simple C-like language in the input to STORESIM.

The 6DOF-RBD solver uses a fully general quaternion formulation to avoid the singularities inherent in the use of the Euler angle formulation. This results in a system of 14 ordinary differential equations governing the motion of the body.

5.1 Six-DOF Rigid Body Dynamics

5.1.1 Basic Equations

The fundamental equations describing the six degrees-of-freedom (6 -dof) dynamics of a rigid body, are expressed with respect to an inertial frame [X,Y,Z], in conjunction with a rotating reference frame $\tilde{X}, \tilde{Y}, \tilde{Z}$, that is fixed to the body (see Figure 5.1). For store simulations where flight durations are relatively short, an earth axis system [X,Y,Z] is a good approximation to an inertial frame. Conversely, for long duration motions, it would be necessary to include the effect of the earths rotation with respect to a star fixed inertial frame. In the present study, it is assumed that [X,Y,Z] is earth fixed.

These fundamental equations, which are in fact expressions of Newtons Laws for the material particles forming the body are:

$$F_{ext} = m * \ddot{r}_0 \tag{5.1}$$

$$M_{ext} = \dot{H}_0 \tag{5.2}$$

where,

m = mass of the body
r_0 = position vector of the bodys mass center "0"
H_0 = angular momentum vector
F_{ext} = external forces arising from aerodynamic and gravitational loads.
M_{ext} = external moments arising from aerodynamic and gravitational loads.

5.1.2 Rates of Change of Linear and Angular Momentum

Equation (5.1) (5.2) relate externally impressed forces (F_{ext} and M_{ext}) and inertia forces developed by the rigid body. As these forces are expressed in the inertial frame [X, Y, Z], it is essential to evaluate the rates of change of linear and angular momentum as seen by an observer [X,Y,Z].

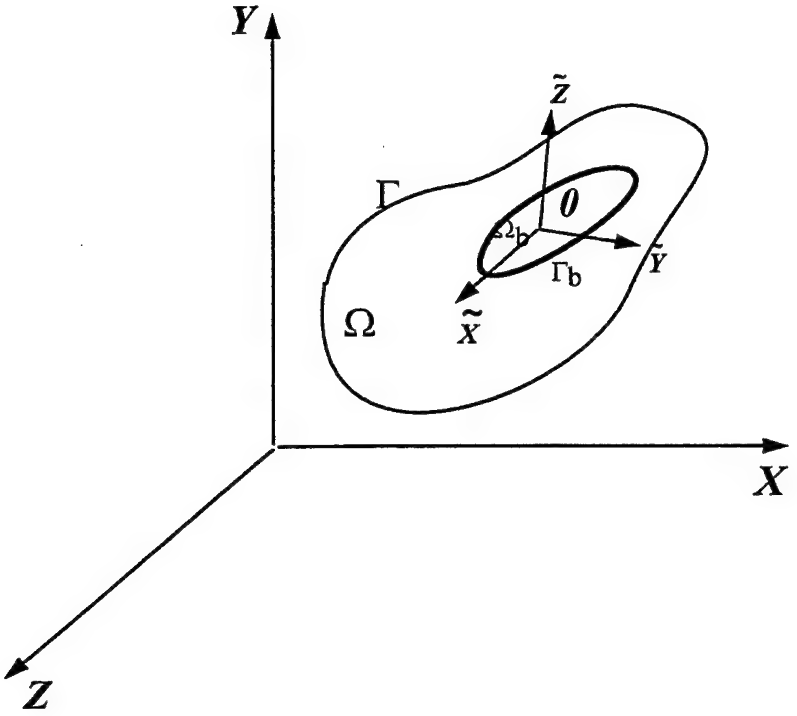


Figure 5.1: Coordinate system for the fluid domain and the moving body.

5.1.3 Rate of Change of Linear Momentum

This is written as $m * \ddot{r}_0$. It is convenient to express the same in terms of the velocity components ($\tilde{u}_0$ of the mass center. The (˜) is used to signify that $\tilde{u}_0$ represents the components of the velocity vector in a co-ordinate system that is instantaneously aligned with the body axes $[\tilde{X}, \tilde{Y}, \tilde{Z}\ ]$. Likewise, $\tilde{F}_{ext}$ represents the components of the external force F_{ext}.

Now, the linear momentum vector as seen by an observer in $[\tilde{X}, \tilde{Y}, \tilde{Z}]$, is $m * \tilde{u}_0$. The rate of change of this vector, as seen by an observer in [X,Y,Z], is obtained with the aid of a standard result governing rotating reference frames, viz:

$$\dot{A}_{xyz} = \dot{A}_{\tilde{x}\tilde{y}\tilde{z}} + \tilde{\omega}_b \otimes A \tag{5.3}$$

where

$$\tilde{\omega}_b = \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} = \textit{rotational velocity of the body.}$$

Thus, using (5.3) we obtain:

$$m\dot{\tilde{u}}_0 = m\dot{\tilde{u}}_0|_{\tilde{x}\tilde{y}\tilde{z}} + \tilde{\omega}_b \otimes m\tilde{u}_0 = \tilde{F}_{ext} \tag{5.4}$$

5.1.4 Rate of Change of Angular Momentum

This is determined in a manner analogous to that used for evaluating the linear momentum. The angular momentum vector seen by an observer in $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ is:

$$H_0|_{\tilde{x},\tilde{y},\tilde{z}} = \tilde{I}\tilde{\omega}_b \tag{5.5}$$

where $\tilde{I}$ is the symmetric inertia tensor of the body in the body axes:

$$\tilde{I} = \begin{bmatrix} I_{\tilde{x}\tilde{x}} & -I_{\tilde{x}\tilde{y}} & -I_{\tilde{x}\tilde{z}} \\ -I_{\tilde{y}\tilde{x}} & I_{\tilde{y}\tilde{y}} & -I_{\tilde{y}\tilde{z}} \\ -I_{\tilde{z}\tilde{x}} & -I_{\tilde{z}\tilde{y}} & I_{\tilde{z}\tilde{z}} \end{bmatrix} \tag{5.6}$$

Thus, as in (5.3), we obtain

$$\begin{aligned} \dot{H}_0|_{xyz} &= \dot{H}_0|_{\tilde{x}\tilde{y}\tilde{z}} + \tilde{\omega}_b \otimes H_0|_{\tilde{x}\tilde{y}\tilde{z}} \\ &= \tilde{I}\dot{\tilde{\omega}}_b + \tilde{\omega}_b \otimes (\tilde{I}\tilde{\omega}_b) = \tilde{M}_{ext} \end{aligned} \tag{5.7}$$

Finally from (5.4) and (5.7) we get

$$\dot{\tilde{u}}_0 = \begin{Bmatrix} \dot{u}_0 \\ \dot{v}_0 \\ \dot{w}_0 \end{Bmatrix} = \left[\frac{\tilde{F}_{ext}}{m} - \tilde{\omega}_b \otimes \tilde{u}_0\right] \tag{5.8}$$

$$\dot{\tilde{\omega}}_b = \begin{Bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{Bmatrix} = \left[\tilde{I}^{-1}\right] * \left[\tilde{M}_{ext} - \tilde{\omega}_b \otimes (\tilde{I}\tilde{\omega}_b)\right] \tag{5.9}$$

5.1.5 Transformation Between the Body Fixed and Inertial Axes Systems

The orientation of the body axes $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ with respect to [X,Y,Z] is needed for two reasons:

- To determine the components of the external forces and moments, as required by the 6-dof equations.
- To determine the precise location of the desired points on the body surface. Note, that the 6-dof equations calculate only the position of the mass center.

Now, in principle, the rotational velocity $\tilde{\omega}_b$ can be integrated in time to obtain the body's orientation. However, it is necessary to express any arbitrary rotation in terms of a sequence of 3 independent rotations. This is done in a standard manner using "Eulerian angles", that we present in the next section, where it will be seen that the time rates of change of the Eulerian angles are in fact related to the rotational velocity. Finally, we present the so called quaternionic variables (Euler parameters) whose use is preferred to the Eulerian angles in numerical computations of 6-dof dynamics.

These are defined by the three elementary rotations ψ (yaw), θ $(pitch)$, and ϕ $(roll)$ which lead the earth fixed axes [X,Y,Z] (see Figure 5.2). $[X_1, Y_1, Z_1]$ and $[X_2, Y_2, Z_2]$ are intermediate positions of the axes at each stage of the Eulerian rotations. These facilitate the formulation of the rotation matrix between the earth and body axes.

5.1.6 Rotation Matrices

The sequence of rotations described by the Eulerian angles leads to the following rotation matrices, G_ψ, G_θ, G_ϕ.

$$\begin{aligned} [x, y, z] &\overset{G_\psi}{\rightarrow} [x_1, y_1, z_1] \\ [x_1, y_1, z_1] &\overset{G_\theta}{\rightarrow} [x_2, y_2, z_2] \\ [x_2, y_2, z_2] &\overset{G_\phi}{\rightarrow} [\tilde{x}, \tilde{y}, \tilde{z}] \end{aligned} \tag{5.10}$$

where

$$G_\phi = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} G_\theta = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}$$

$$G_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix}$$

Thus, $[X, Y, Z] \overset{G}{\rightarrow} [\tilde{X}, \tilde{Y}, \tilde{Z}]$ where

$$G = G_\phi \cdot G_\theta \cdot G_\psi \tag{5.11}$$

Note that

$$\begin{aligned} G_\psi^{-1} = G_\psi^T &: G_\theta^{-1} = G_\theta^T \\ G_\phi^{-1} = G_\phi^T &: G^{-1} = G^T \end{aligned} \tag{5.12}$$

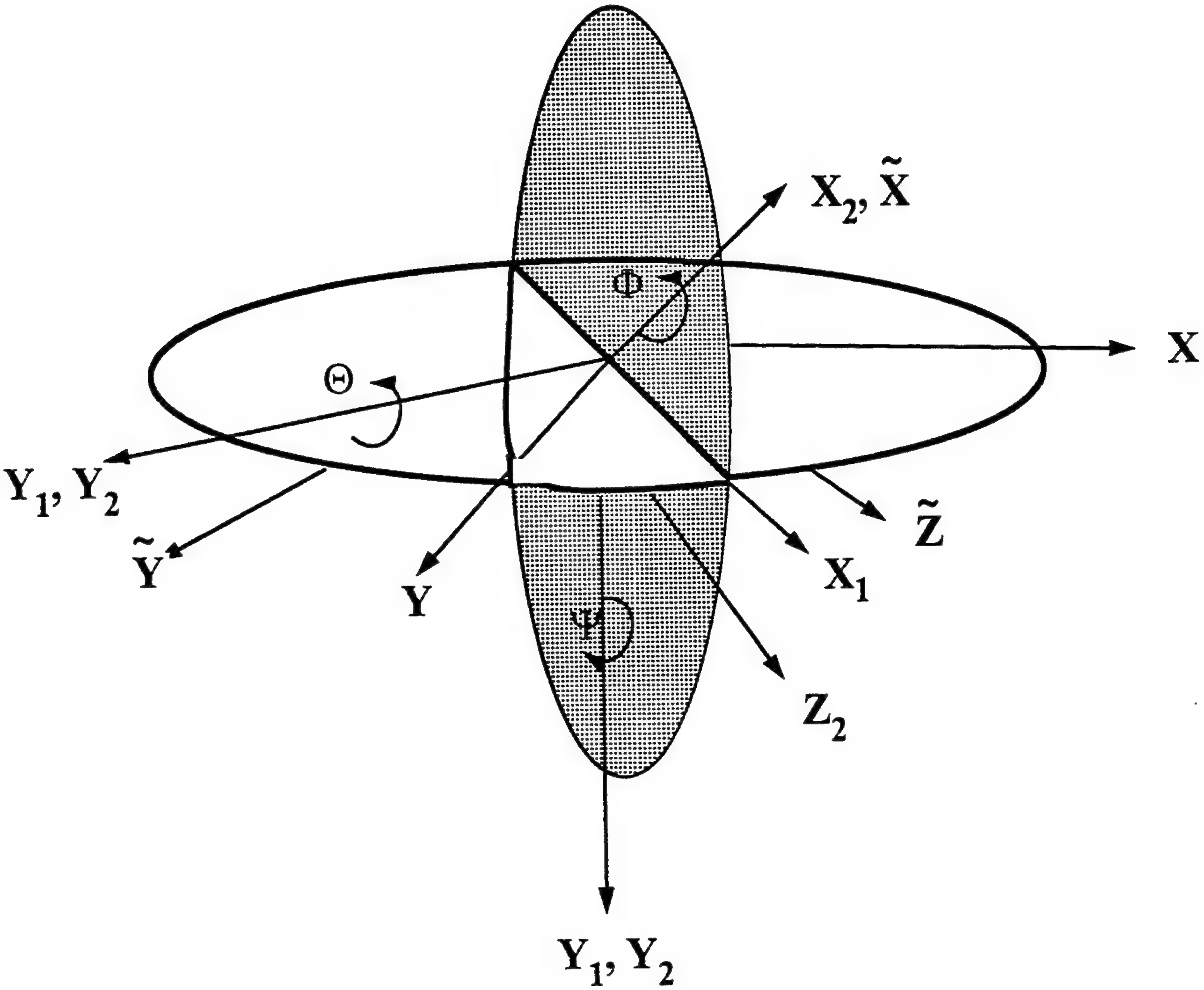


Figure 5.2: Coordinate Axes

5.1.7 Relation Between Rotational Velocity and the Time Derivatives of the Euler Angles

The rotational velocity of the body, ω_b has components [p,q,r] in the body axes system. Now, stemming from the definition of the Euler angles, we can also express ω_b in terms of its components along the directions of the three elementary rotations. Thus, we obtain,

$$\omega_b \equiv \dot{\psi} \cdot n_{z_1} + \dot{\theta}_{y_2} + \dot{\phi} \cdot n_{\tilde{x}} \tag{5.13}$$

where $n_{z_1}, n_{y_2}, n_{\tilde{x}}$ are unit vectors along directions $Z1, Y2, \tilde{X}$ respectively.

Using (5.13) in conjunction with (5.10) one can show that

$$\left\{ \begin{array}{l} \dot{\phi} - \dot{\psi} sin\theta \\ \dot{\psi} cos\theta \cdot sin\phi + \dot{\theta} cos\phi \\ \dot{\psi} cos\theta cos\phi - \dot{\theta} sin\phi \end{array} \right\} = \left\{ \begin{array}{c} p \\ q \\ r \end{array} \right\} \tag{5.14}$$

It is clear from (5.14) that p,q,r can be integrated with respect to time to give ψ, θ, ϕ. However, as we shall see in the next section, the Eulerian angles are not the most suitable parameters to use for rigid body dynamics, and that it is preferable to employ the so called “quarternionic variables“(Euler parameters) in their place.

5.1.8 Singular Behavior of the Rotational Derivatives

From (5.14), we see that the time derivative of the Euler angles are related to the rotational velocity components by the relation:

$$[\tilde{E}] \left\{ \begin{array}{c} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{array} \right\} = \left\{ \begin{array}{c} p \\ q \\ r \end{array} \right\} \tag{5.15}$$

where

$$\tilde{E} = \begin{bmatrix} -sin\theta & 0 & 1 \\ cos\theta \cdot sin\phi & cos\phi & 0 \\ cos\theta \cdot cos\phi & -sin\phi & 0 \end{bmatrix} \tag{5.16}$$

Now $Det(\tilde{E}) = -cos\theta = 0$ (if $\theta = \pm\pi/2$) Hence, at $\theta = \pm\pi/2$, equation (5.14) is singular. Geometrically, this stems from the fact that at $\theta = \pm\pi/2$, ψ is undefined and can take any value. The effect of this singularity on numerical computations is to cause increasely large errors in the vicinity of $\theta = \pm\pi/2$, leading to completely erroneous results. The standard remedy employed in this case is to use “quarternionic variables ”.

5.2 Quarternions (Euler Parameters)

Any arbitrary rotation in 3-D space is equivalent to a pure finite rotation (α) about a uniquely defined axes. This corresponds to the mathematical fact that the rotation matrix governing a rotation has a single real eigen value "λ" and a unique eigenvector (with normalized components n_i). This is equivalent to a pure rotation of $\alpha = \lambda$ along the axis $n - i$.

The four quarternions are then defined as follows:

$$\hat{q}_i = n_i sin\alpha/2 \ : \ i = 1, 2, 3$$

$$\hat{q}_4 = cos\alpha/2 \tag{5.17}$$

It is easily verified that

$$\hat{q}_1^2 + \hat{q}_2^2 + \hat{q}_3^2 + \hat{q}_4^2 = 1 \tag{5.18}$$

The rotation matrix can be expressed in terms of the quarternions as:

$$G = \begin{bmatrix} 2(\hat{q}_1^2 + \hat{q}_4^2) - 1 & 2(\hat{q}_1\hat{q}_2 + \hat{q}_3\hat{q}_4) & 2(\hat{q}_1\hat{q}_3 - \hat{q}_2\hat{q}_4) \\ 2(\hat{q}_1\hat{q}_2 - \hat{q}_3\hat{q}_4) & 2(\hat{q}_2^2 + \hat{q}_4^2) - 1 & 2(\hat{q}_2\hat{q}_3 + \hat{q}_1\hat{q}_4) \\ 2(\hat{q}_1\hat{q}_3 + \hat{q}_2\hat{q}_4) & 2(\hat{q}_2\hat{q}_3 - \hat{q}_1\hat{q}_4) & 2(\hat{q}_3^2 + \hat{q}_4^2) - 1 \end{bmatrix} \tag{5.19}$$

Also, the time derivatives of the quarternions are related to the rotational rates by:

$$\begin{Bmatrix} \dot{\hat{q}}_1 \\ \dot{\hat{q}}_2 \\ \dot{\hat{q}}_3 \\ \dot{\hat{q}}_4 \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 & -\hat{q}_1 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 & -\hat{q}_2 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 & -\hat{q}_3 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 & -\hat{q}_4 \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \\ 0 \end{Bmatrix} \tag{5.20}$$

Now, it is evident from (5.20) that the time derivatives of the quarternions are always uniquely defined. This is contrary to that for the Eulerian angles, which have a singularity at $\theta = \pm\pi/2$. Thus, (5.20) is used instead of (5.14) in rigid body dynamics. The Euler angles are used only to specify an initial orientation for the body. The next section contains formulae necessary to determine quarternions knowing the Euler angles and vice versa.

5.2.1 Quarternions in Terms of Euler Angles

Let G_{ij} be the coefficients of the rotation matrix G. The quarternions are then:

$$\hat{q}_4 = 0.5 * [1 + G_{11} + G_{22} + G_{33}]^{\frac{1}{2}} \tag{5.21}$$

$$\left.\begin{aligned} \hat{q}_1 &= (G_{23} - G_{32})/4\hat{q}_4 \\ \hat{q}_2 &= (G_{31} - G_{13})/4\hat{q}_4 \\ \hat{q}_3 &= (G_{12} - G_{21})/4\hat{q}_4 \end{aligned}\right\} \tag{5.22}$$

For the special case where $\hat{q}_4 = 0$, the following logic is used:

```
$f_i = 1 + G_{ii}$        $: i = 1, 2, 3$

if ($f_1 \neq 0$)         then
                          $\hat{q}_1 = (0.5 * f_1)^{\frac{1}{2}}$ : $\hat{q}_2 = 0.5 * G_{12}/\hat{q}_1$ : $\hat{q}_3 = 0.5 * G_{13}/\hat{q}_1$
else if ($f_2 \neq 0$)    then
                          $\hat{q}_2 = (0.5 * f_2)^{\frac{1}{2}}$ : $\hat{q}_1 = 0.5 * G_{21}/\hat{q}_2$ : $\hat{q}_3 = 0.5 * G_{23}/\hat{q}_2$
else if ($f_3 \neq 0$)    then
                          $\hat{q}_3 = (0.5 * f_3)^{\frac{1}{2}}$ : $\hat{q}_1 = 0.5 * G_{31}/\hat{q}_3$ : $\hat{q}_2 = 0.5 * G_{32}/\hat{q}_3$
end if
```

5.2.2 Euler Angles in Terms of Quarternions

$$\sin\theta = 2(\hat{q}_2\hat{q}_4 - \hat{q}_1\hat{q}_3) \tag{5.23}$$

There is no loss of generality in restricting θ to the range $-\pi/2 \leq \theta \leq \pi/2$.

$$tan\psi = \frac{2(\hat{q}_1\hat{q}_2 + \hat{q}_3\hat{q}_4)}{2(\hat{q}_1^2 + \hat{q}_4^2) - 1} \tag{5.24}$$

$$tan\phi = \frac{2(\hat{q}_2\hat{q}_3 + \hat{q}_1\hat{q}_4)}{2(\hat{q}_1^2 + \hat{q}_4^2) - 1} \tag{5.25}$$

ψ and ϕ should be determined by the ATAN2 function, to obtain values in the correct quatrant.

5.3 The Full Equations of Rigid Body Dynamics

Equation (5.8), (5.9), and (5.20) constitute a set of 10 simultaneous *odes* (ordinary differential equations) in time. The addition of three more equations governing the position vector of the mass center,viz.

$$\dot{\boldsymbol{r}}_0 = \boldsymbol{G}^T\tilde{\boldsymbol{u}}_0 \tag{5.26}$$

completes the set of equations that are needed to determine the trajectory of the body. These equations are solved as an initial value problem using Runge-Kutta timestepping.

5.4 Store Simulation

5.4.1 Problem Statement

- Let $[X, Y, Z]$ be an inertial frame (see Figure 5.1). This could, for example, be fixed to an aircraft flying along a steady, straight path.
- Let Γ be a domain representing the farstream, fixed to $[X, Y, Z]$.
- Let Ω_b be a body whose inertial orientation and velocity with respect to $[X, Y, Z]$ is known. $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ is a local axes system attached to the body.

5.4.2 Solution Strategy

First, it is necessary to specify a set of independent parameters which compose the solution set. These are:

$\boldsymbol{U}$	The vector of conservation flow variables at each grid point.
$\boldsymbol{U}_{gflow}$	The grid velocity components at each point in the interior of the mesh.
$\boldsymbol{X}_{gflow}$	The grid coordinates of interior nodes.
$\hat{\boldsymbol{q}}$	Quaternions governing orientation of $[\tilde{X}, \tilde{Y}, \tilde{Z}]$.
$\tilde{\boldsymbol{u}}_0 = (\tilde{u}_0, \tilde{v}_0, \tilde{w}_0)$	Velocity components of the mass center "0" of the body in the $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ system.
$[X_0, Y_0, Z_0] = \boldsymbol{r}_0$	Coordinates of "0" in $[X, Y, Z]$.
$(\tilde{p}, \tilde{q}, \tilde{r}) = \tilde{\omega}_b$	Rotational velocity of the body in the $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ system.
$\tilde{\boldsymbol{r}}_b$	Coordinates of body nodes in the $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ system.

We consider the following two problems:

(a) Given the mass properties of Ω_b, compute its trajectory under the influence of external forces (such as gravitational and aerodynamic).

(b) Given the trajectory Ω_b, compute the aerodynamic forces acting on it.

Next we define the following derived quantities that are important to the solution methodology.

$\tilde{u}_{gbody}$	:	The grid velocity components of nodes on the body in $[\tilde{X}, \tilde{Y}, \tilde{Z}]$ system.
	$=$	$\tilde{\boldsymbol{u}}_0 + \tilde{\omega}_b \bigoplus \tilde{\boldsymbol{r}}_b$
$\boldsymbol{U}_{gbody}$	:	Grid velocity of nodes on the body in $[X, Y, Z]$. ,
	$=$	$\boldsymbol{G}^{-1}(\hat{\boldsymbol{q}})u_{g\tilde{body}}$
$\boldsymbol{X}_{body}$	:	Function of $(\tilde{\boldsymbol{r}}b, \hat{\boldsymbol{q}}, \boldsymbol{r}_0)$
	$=$	$\boldsymbol{r}_0 + \boldsymbol{G}^{-1}(\hat{\boldsymbol{q}})\tilde{\boldsymbol{r}}_b$

where $\boldsymbol{G}$ is the rotation matrix relatin $\boldsymbol{X}$ to $\tilde{\boldsymbol{X}}$ by $\tilde{\boldsymbol{X}} = \boldsymbol{G}(\hat{\boldsymbol{q}})\ \boldsymbol{X}$.

At each node point in the mesh, the system of equations that are solved for the flow solver, mesh mover, and six-dof solver are:

Flow solver:

$$\dot{\boldsymbol{U}}^n = \boldsymbol{F}(\boldsymbol{U}^n, \boldsymbol{U}^n_{gbody}, \boldsymbol{U}^n_{gflow}, \boldsymbol{X}^n) \tag{5.27}$$

Mesh mover:

$$\begin{aligned} \dot{\boldsymbol{U}}^n_{gflow} &= \boldsymbol{M}(\boldsymbol{U}^n_{gbody}, \boldsymbol{U}^n_{gflow}, \boldsymbol{X}^n) \\ \dot{\boldsymbol{X}}^n_{gflow} &= \boldsymbol{U}^n_{gflow} \end{aligned} \tag{5.28}$$

Six-DOF solver:

$$\left[\dot{\tilde{\boldsymbol{u}}}_0, \dot{\tilde{\omega}}_0, \dot{\boldsymbol{r}}_0, \dot{\tilde{\boldsymbol{q}}}_0\right]^T = \boldsymbol{S}(\tilde{\boldsymbol{u}}_0, \tilde{\omega}_0, \boldsymbol{r}_0, \tilde{\boldsymbol{q}}_0, \boldsymbol{f}_{ext}, \boldsymbol{m}_{ext}) \tag{5.29}$$

where $\boldsymbol{f}_{ext}, \boldsymbol{m}_{ext}$ are the external forces and moments respectively. Equations (5.27), (5.28), and (5.29) constitute a set of equations, explicit in time, that can be updated by forward Euler or Runge-Kutta time stepping.

If we are interested in solving only the problem of specified trajectory $\boldsymbol{T}(t)$, only equations (5.1) and (5.2) need to be solved. In addition we have:

$$\left[\dot{\boldsymbol{r}}^n_0, \dot{\boldsymbol{U}}^n_0, \tilde{\boldsymbol{q}}^n_0, \tilde{\omega}^n_0\right]^T = \boldsymbol{T}(t) \tag{5.30}$$

where t is the current time value. The body grid velocity and the new position can them be computed from the following:

$$\begin{aligned} \tilde{\boldsymbol{u}}^n_0 &= \boldsymbol{G}(\hat{\boldsymbol{q}}^n)\boldsymbol{U}^n_0 \\ \tilde{\boldsymbol{u}}^n_{gbody} &= \tilde{\boldsymbol{u}}^n_0 + \tilde{\omega}^n_0 \oplus \boldsymbol{r}_b \\ \boldsymbol{U}^n_{gbody} &= \boldsymbol{G}^{-1}(\hat{\boldsymbol{q}}^n)\tilde{\boldsymbol{u}}^n_{gbody} \\ \boldsymbol{X}^n_{body} &= \boldsymbol{r}^n_0 + \boldsymbol{G}^{-1}(\hat{\boldsymbol{q}}^n)\tilde{\boldsymbol{r}}_b \end{aligned}$$

5.5 Node Movement

As the prescribed bodies move (either under captive or calculated trajectories), the mesh around the bodies must be updated to avoid grid tangling. This is accomplished in two steps: 1) grid node movement and 2) mesh restructuring. To achieve movement of interior nodal points in response to the motion of the grid boundary, we appeal to an incompressible flow analogy. The advantage of this method is that it allows the nodes to flow through the domain and around the body as it moved through the field[16].

Assume that each grid node represents a particle of mass (the mass can be different for each node). We define the 'density' of the fluid as the mass of the node divided by the volume of all connecting elements. Initially, the density is assumed to be unity (since each node's mass is exactly equal to the volume of the surrounding elements). We then require that the fluid is incompressible and thus the density remains constant. Interior nodes move in response to moving boundaries according to grid velocities calculated by:

$$\boldsymbol{v}_i = \beta \nabla \rho_i \tag{5.31}$$

where β is a parameter calculated to maintain overall stability of the grid movement algorithm. Given the grid velocities, actual node movement proceeds by integration of the following equation:

$$\frac{\partial \boldsymbol{x}_i}{\partial t} = \boldsymbol{v}_i \tag{5.32}$$

The node movement is done using an analogy with incompressible flow that ensures that the mesh moves consistently with the bodies, and that an acceptable mesh spacing is maintained. The node movement strategy, since it is almost identical to the flow solver methodology, is fully vectorized in the same manner as the flow solver.

5.6 Mesh Restructuring

After several time steps, it may be necessary to restructure the mesh by keeping the node positions fixed, but changing the mesh connectivity. We recover a Delaunay mesh after node movement by performing the 3D edge swapping procedure outlined above. First we update the node locations using the grid node-flow analogy. Next, we visit groups of five nodes that form either two or three tetrahedra and swap the edges where the Delaunay criterion is violated. This method has been implemented very efficiently by keeping a hash table of edges in the mesh that have exactly three surrounding tets (and are candidates for swapping into two new tets).

6 Visualization

Visualization is a key piece of any three-dimensional CFD simulation, and STORESIM provides a full suite of capabilities to visualize the time-dependent, 3D solutions. The visualization module of STORESIM is called STOREVIS. Users can select any time value to be visualized (as long as it was saved into the restart file). If the time value does not exactly correspond to a value in the file, the solution values are linearly interpolated from the neighboring values, transparently to the user. Once a time value is chosen, the user can look at (transparent) planar slices, (transparent) iso-surface, particle traces, and boundary contours of various quantity. The mesh at the given time level can also be viewed.

The visualization package enables the user to inspect the grid generation process interactively. It also enables the user to display selected regions for critical inspection and modification of the input data. This is especially useful during mesh generation as it provides the user an option to change the data without editing the input files.

The post-processing module enables the user to display multiple slices of the flow domain. It also allows the user to plot the selected variables on an isosurface of a different variable. For example, the user can plot the variation of density on an iso-surface of the pressure. The velocity vectors and particle traces option allow the user to plot the flow field and understand the flow behavior in various regions of the domain.

7 Results and Discussion

7.1 Preliminary Remarks

In the previous sections we have described important features of the mesh generation, flow solver, six-dof solver, and visualization modules of STORESIM. In this section we present a number of validation and demonstration problems which demonstrate the utility and flexibility of the package. In each case all of the modules are used. In other words, we start with the geometry definition, then generate an unstructured mesh using TETMESH. Next, we use the integrated flow solver STOREFLOW (including the six-dof solver and mesh mover modules when appropriate) to obtain the flow solution. Finally, the results are post-processed using STOREVIS. The problems have a varying degree of complexity in terms of both mesh generation and flow analysis. These problems have been selected to test all of the major features of the software.

7.2 Validation Examples

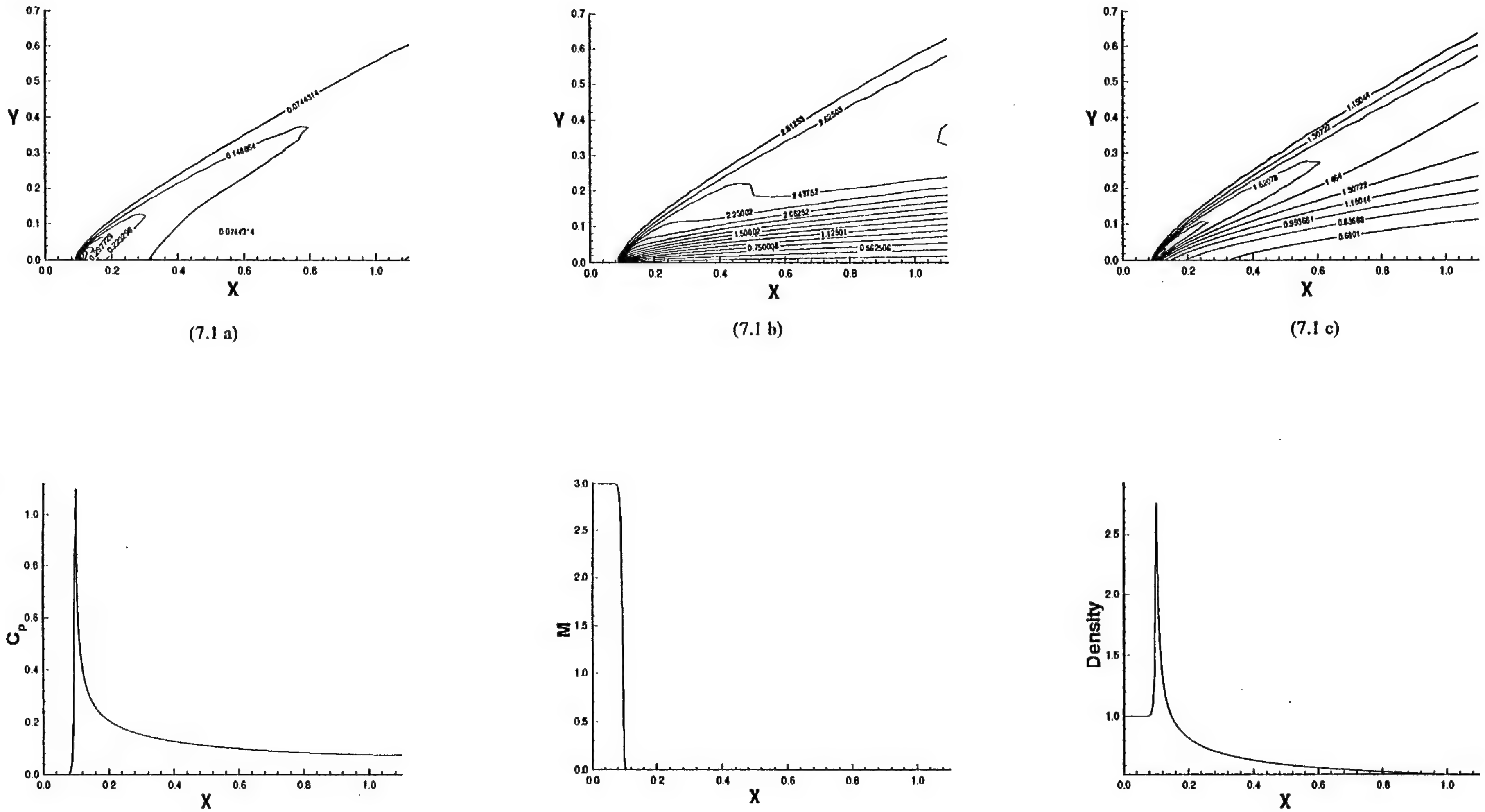
7.2.1 Carter Problem

First we solve flow over a flat plate. The free stream Mach number of the flow is 3.0 and the Reynolds numbers is 1000, and the temperature of the plate is held at $1000K$. This problem is solved to validate the viscous flow solver. The problem is modeled using a 1800 quadrilateral elements with element biasing towards the wall. Figure 7.1 shows the [ressure coefficient C_p, Mach number, and density variations along the solid wall and in the flow domain. The Mach contours show a boundary layer formation at the isothermal solid wall. Also this indicates the flow causes a curved shock at the leading edge of the plage. These results agree with the values presented in [20].

Next we increase the Reynolds number to $1E+06$ and solved the problem using the zero-equation turbulence model. Computed values of C_p, Mach number and density are plotted in Figure 7.2. Comparing these with the laminar flow solution indicates that the wall boundary layer for turbulent flow, as expected, is much thinner than the laminar boundary layer.

7.2.2 Supersonic Flow Over a Sphere

As a second test case, we consider the problem of supersonic inviscid flow over a sphere. The radius of the sphere is 1 unit. The radius of the outer (farfield) boundary is assumed to be 10 units. The free stream Mach number is assumed to be 3.0. The inner and outer boundaries are each represented by eight network patches as shown in Figure 7.3. Each of these 16 patches has one degenerate edge. To produce a similar triangulation on each surface, we discretized each edge using 15 nodes. This edge discretization produced 188 triangles on each surface and 3008 triangles on all 16 surfaces. Surface triangulation on the inner boundary is shown in Figure 7.4. Next we extruded the inner surface by 0.25 units to produce 5 layers of prismatic elements. The prismatic elements are shown in Figure 7.5 The final mesh consists of 17,156



(7.1 a)

(7.1 b)

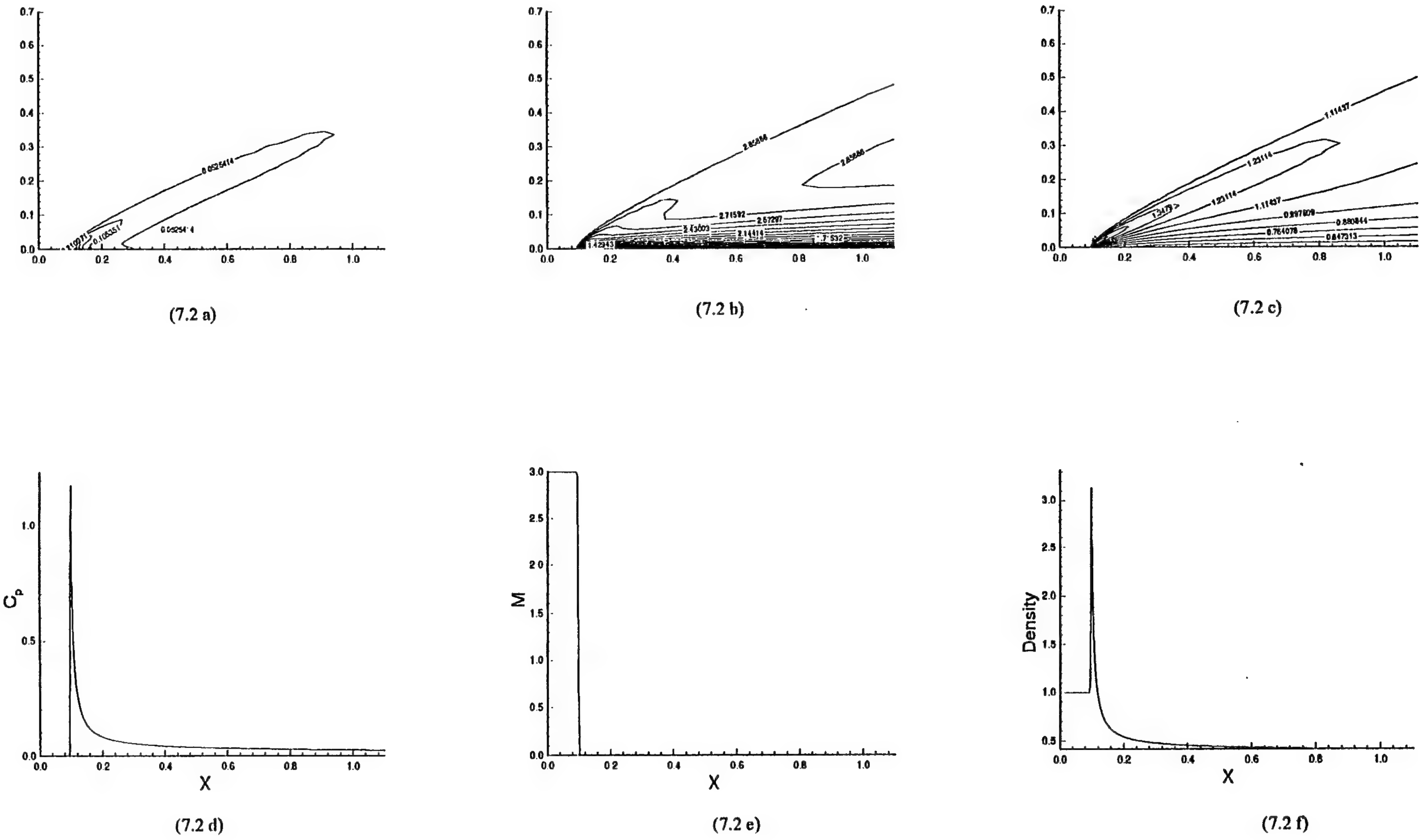
(7.1 c)

(7.1 d)

(7.1 e)

(7.1 f)

Figure 7.1: Laminar compressible fluid flow over a flat a plate (Re = 1000)
Plots of pressure coefficient, Mach number and density.



(7.2 a)

(7.2 b)

(7.2 c)

(7.2 d)

(7.2 e)

(7.2 f)

Figure 7.2: Trubulent compressible fluid flow over a flat a plate (Re = 1.0E+06)
Plots of pressure coefficient, Mach number and density.

nodes, 80,406 tetrahedral elements and 7520 prisms. The complete mesh generation process required 600 seconds on SGI Indigo r4000 running IRIX 5.3.

Here we present the results for inviscid flow analysis. Mach number contours (see Figure 7.6a) show the formation of a bow shock ahead of the sphere. The standoff distance and the stagnation properties are in good agreement with the experimental values. Velocity vectors (Figure 7.6b) at the midplane of the sphere in the flow direction show a recirculation region behind the sphere. Pressure contours and density contours are shown in Figure 7.6c and 7.6d respectively.

7.2.3 Double Ellipsoid

Next we consider hypersonic flow over double ellipsoid. Surface geometry of the double ellipsoid is given in Figure 7.7. The free stream Mach number for this problem is 8.15 and the angle of attack is 30 deg. Figure 7.8 shows the trimmed surface and their normals, edges of the trimmed surface, and the edge discretization. The node distribution along each edge is shown in green circles. Figure 7.9 shows the wireframe representation of the double ellipsoid surface. This figure shows the cusped narrow regions where the top and bottom ellipsoids meet. Such surfaces are especially challenging to automatic mesh generators because of the surface curvatures and non-unique normals. Figures 7.10 and 7.11 show the surface triangulation. A total of 15,794 surface triangles were created on both double ellipsoid surface and the farstream bounding box. After volume meshing the final mesh had 65,608 nodes and 392,921 tetrahedral elements. The complete mesh generation process (including triangulation and tetrahedralization) required 5400 seconds on SGI Indigo r4000 running IRIX 5.3.

The free stream Mach number and the angle of attack are taken to be equal to 8.15 and 30 deg, respectively. Inviscid flow analysis results are compared with those presented in [7]. Figure 7.12 shows the contours of density, and coefficient of pressure on the double ellipsoid surface. Also, plotted in Figure 7.12 are variation of density and pressure coefficient along the symmetry plane. Computed values are compared with those presented in [7] and they show excellent agreement. Figure 7.13 shows the Mach contours at the symmetry plane. These contours show a bow shock formation ahead of the nose of the ellipsoid and another shock formation at the intersection of the two ellipsoids.

7.2.4 Flow over a 3-D Obstacle

Here we consider the problem of flow over a three-dimensional obstacle. A Schematic diagram of the flow domain is shown in Figure 7.14. Surface triangulation is shown in Figure 7.15. The swept angle of the obstacle is 30 deg and the plates are inclined at an angle or 15 deg. The free stream Mach number and Reynolds number are assumed to be equal to 10 and 9.0E+06, respectively. This flow is modeled as laminar nonreacting flow. The mesh consists of 10 138 surface triangles, 25 619 nodes, and 146 362 tetrahedral elements. The complete mesh generation process required 2200 seconds on SGI Indigo r4000 running IRIX 5.3. Figure 7.16 shows the velocity at the mid plane. Figure 7.17 shows the Mach contours.

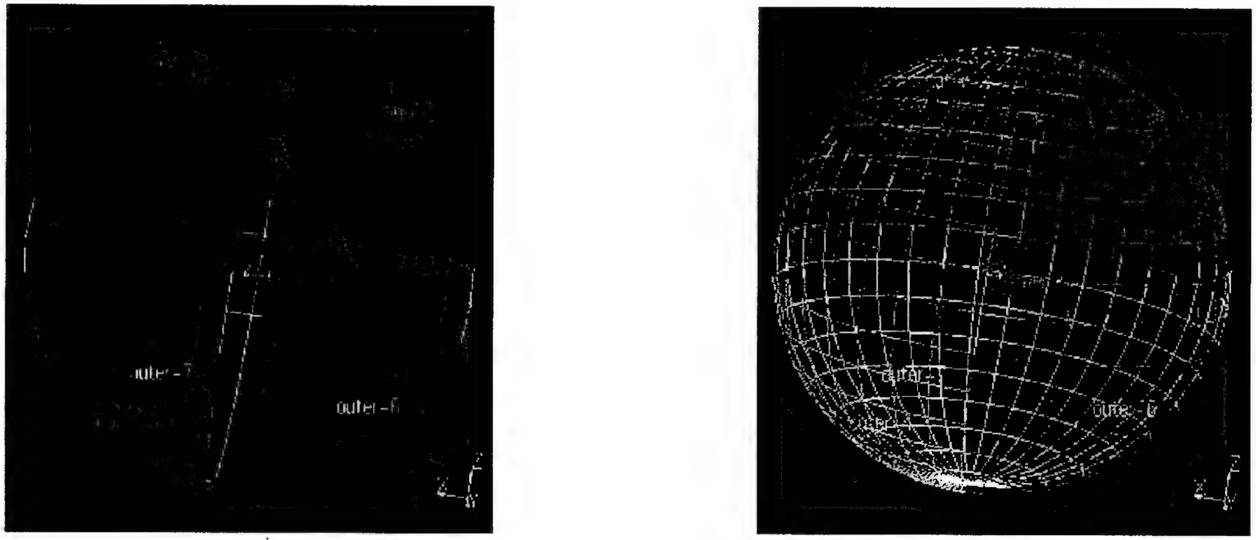


Figure 7.3: Surface names and wireframe plot of the network patches of inner and outer spheres.

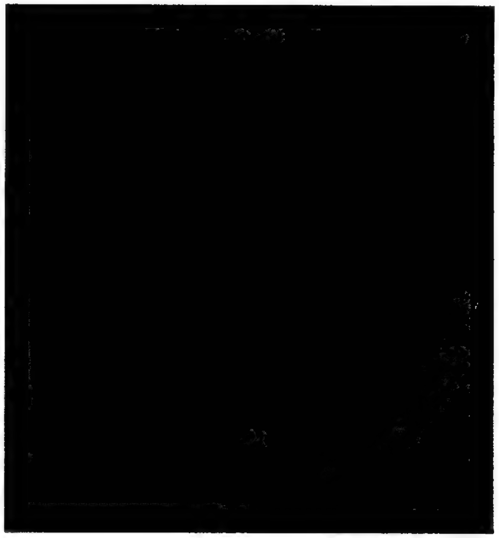


Figure 7.4: Surface triangulation

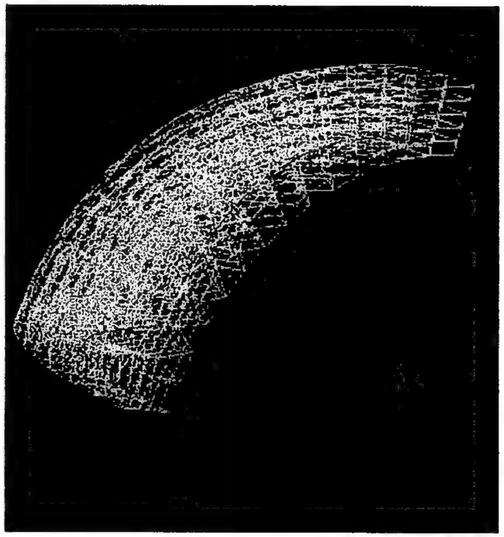
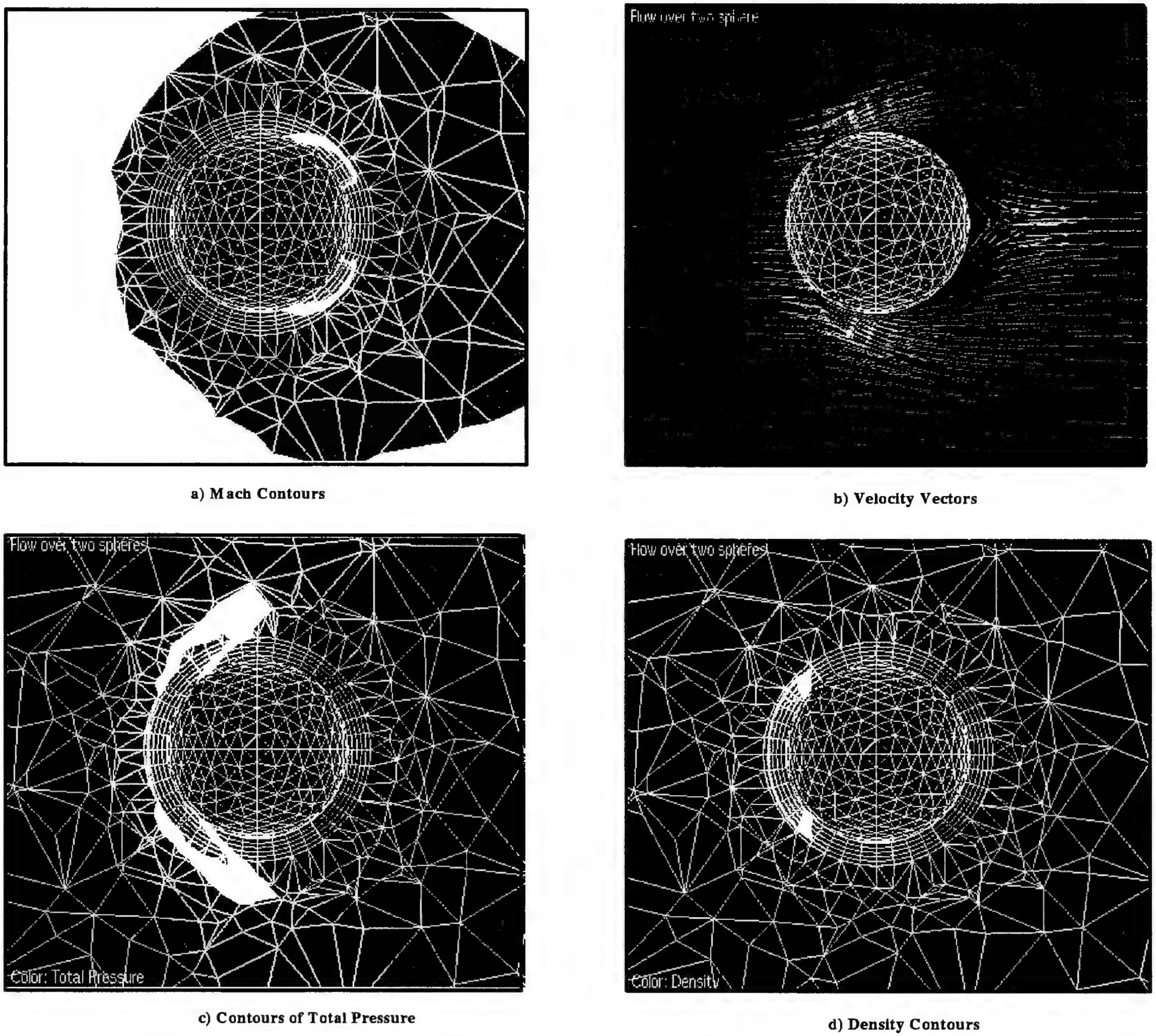


Figure 7.5: Prismatic elements on one patch.



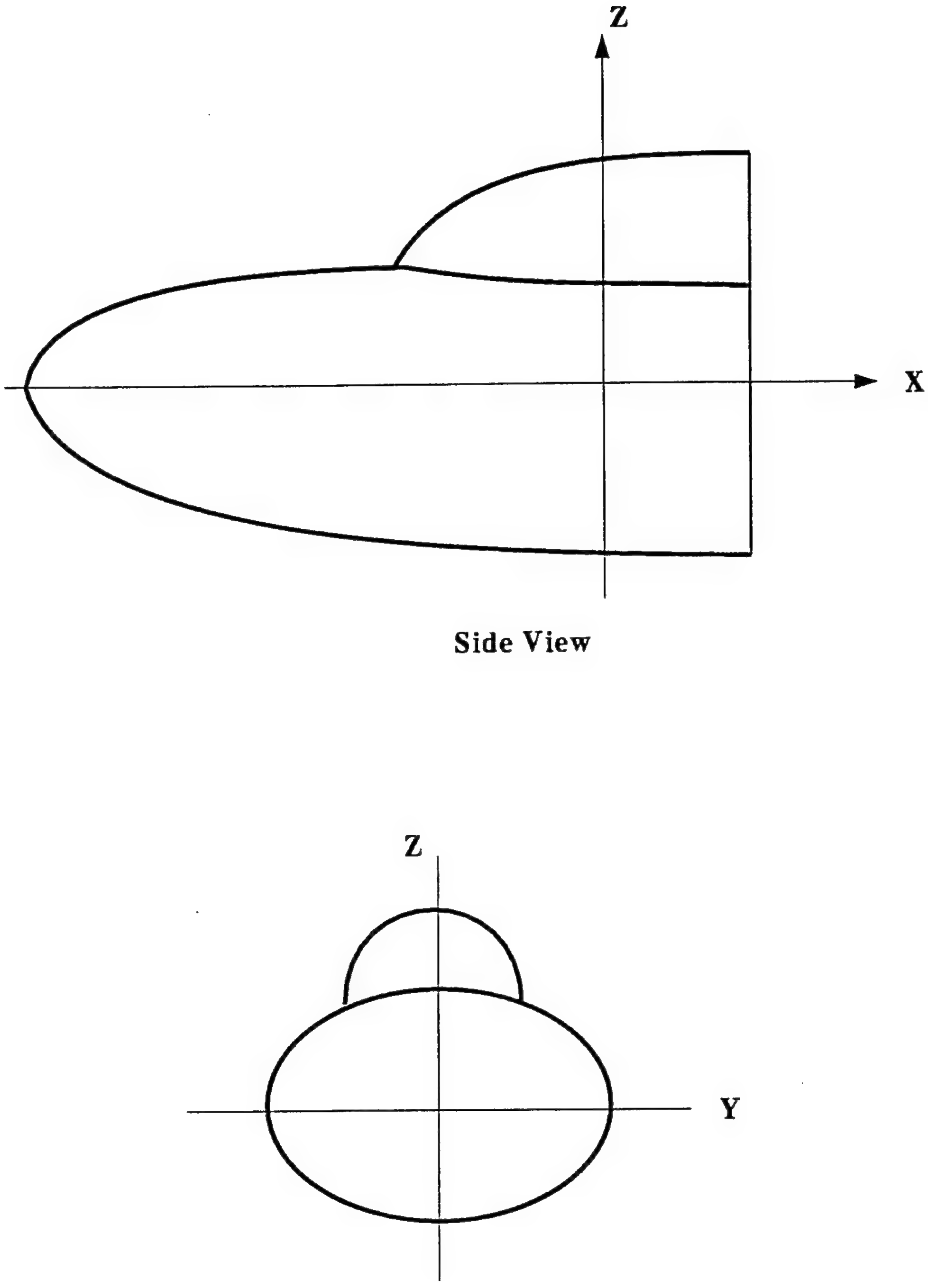
a) Mach Contours

b) Velocity Vectors

c) Contours of Total Pressure

d) Density Contours

Figure 7.6: Flow over a sphere at M =3. Velocity vectors and contours of Mach number, density, and total pressure are plotted on the symmetry plane in flow direction



Side View

Front View

Figure 7.7: Schematic Diagram of Double Ellipsoid Geometry.

Figure 7.8: Trimmed surfaces, surface normals, edge normals, and edge discretization for double ellipsoid geometry.

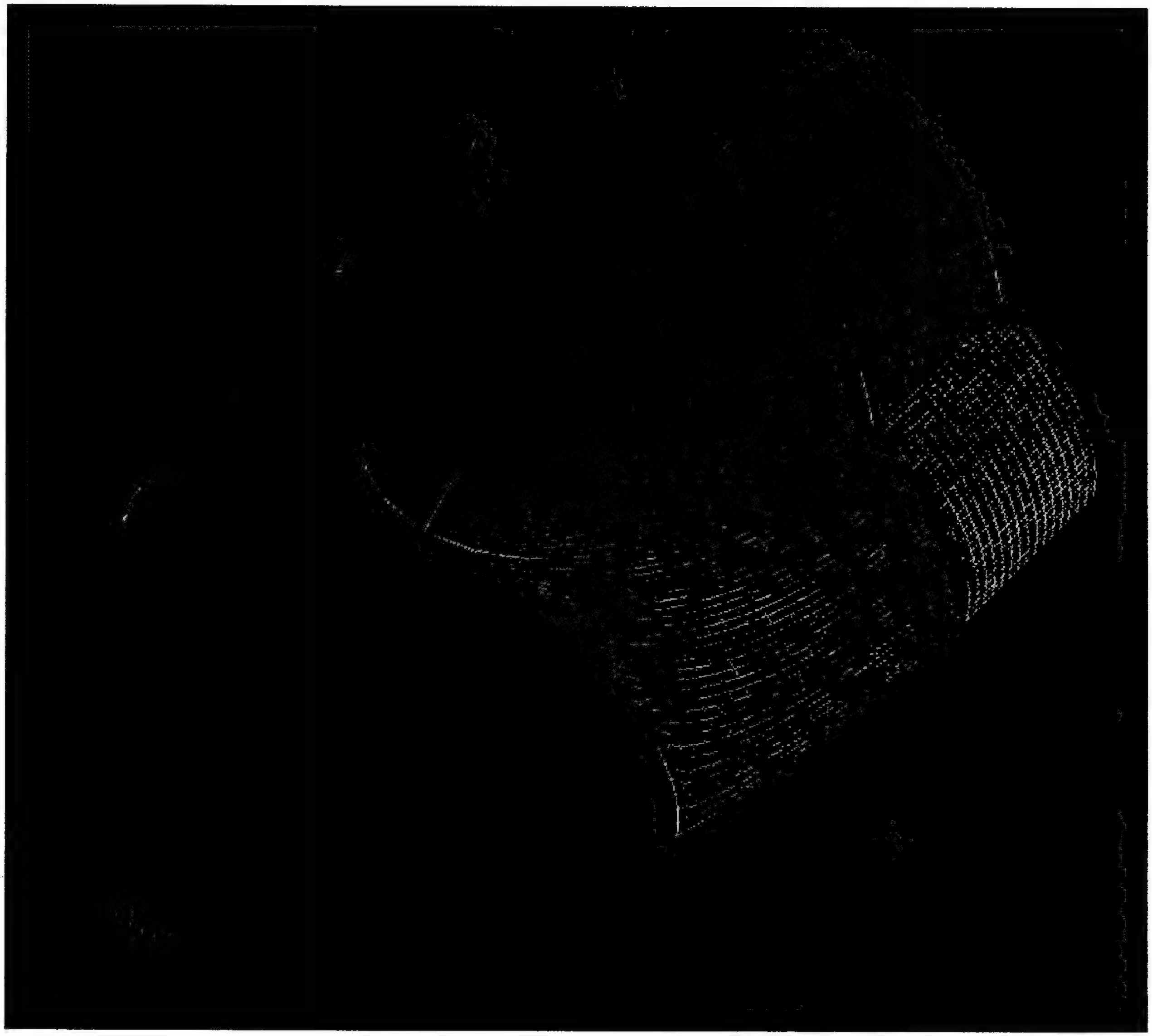


Figure 7.9: Wireframe representation of the double ellipsoid surface. This is used as the input and the surface triangles are created such that the nodes fall on this surface.

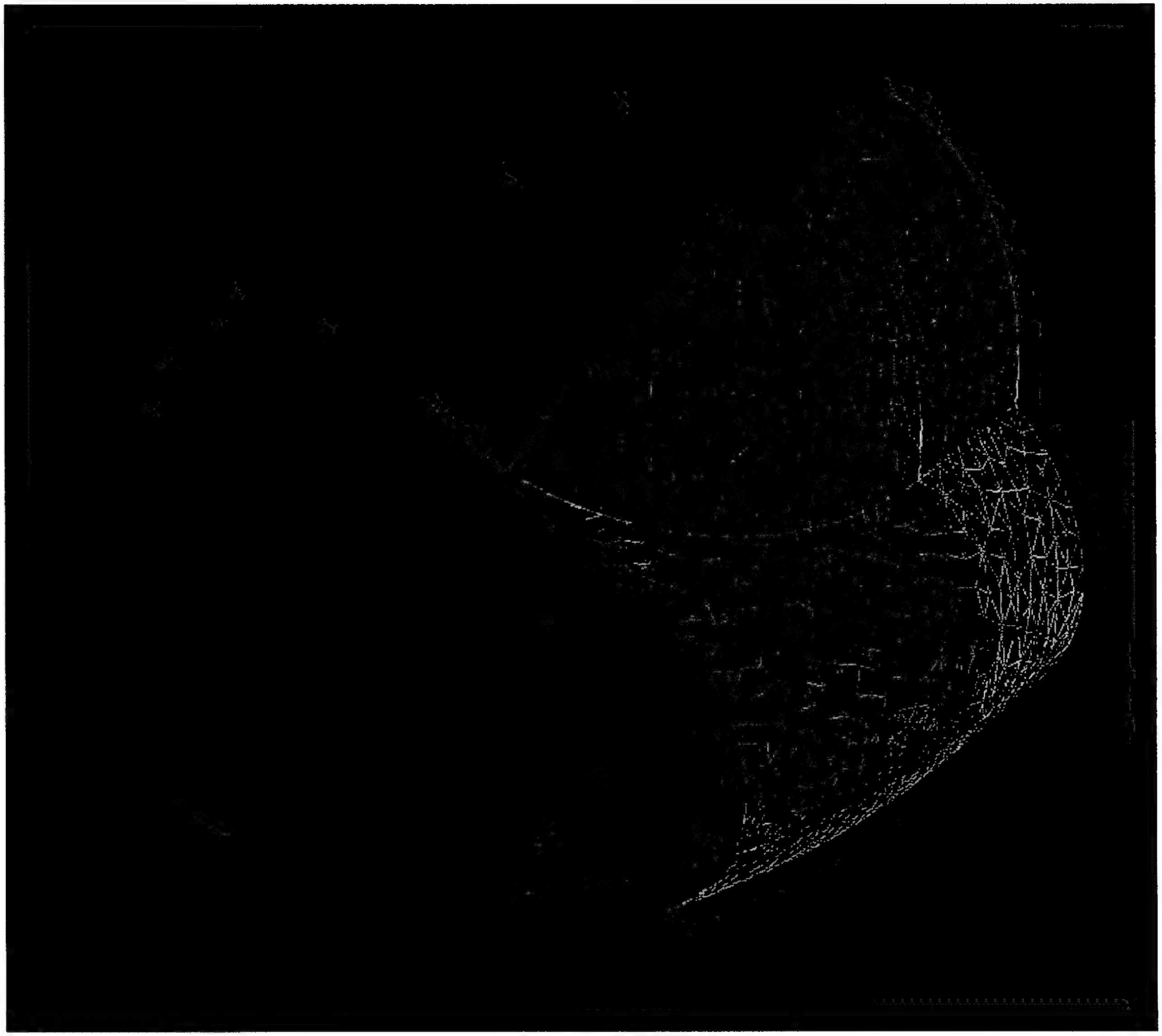


Figure 7.10: Triangulation of the surface of the double ellipsoid. Total number (both on outer boundary and the double ellipsoid surface) of triangles created is equal to 15,794.

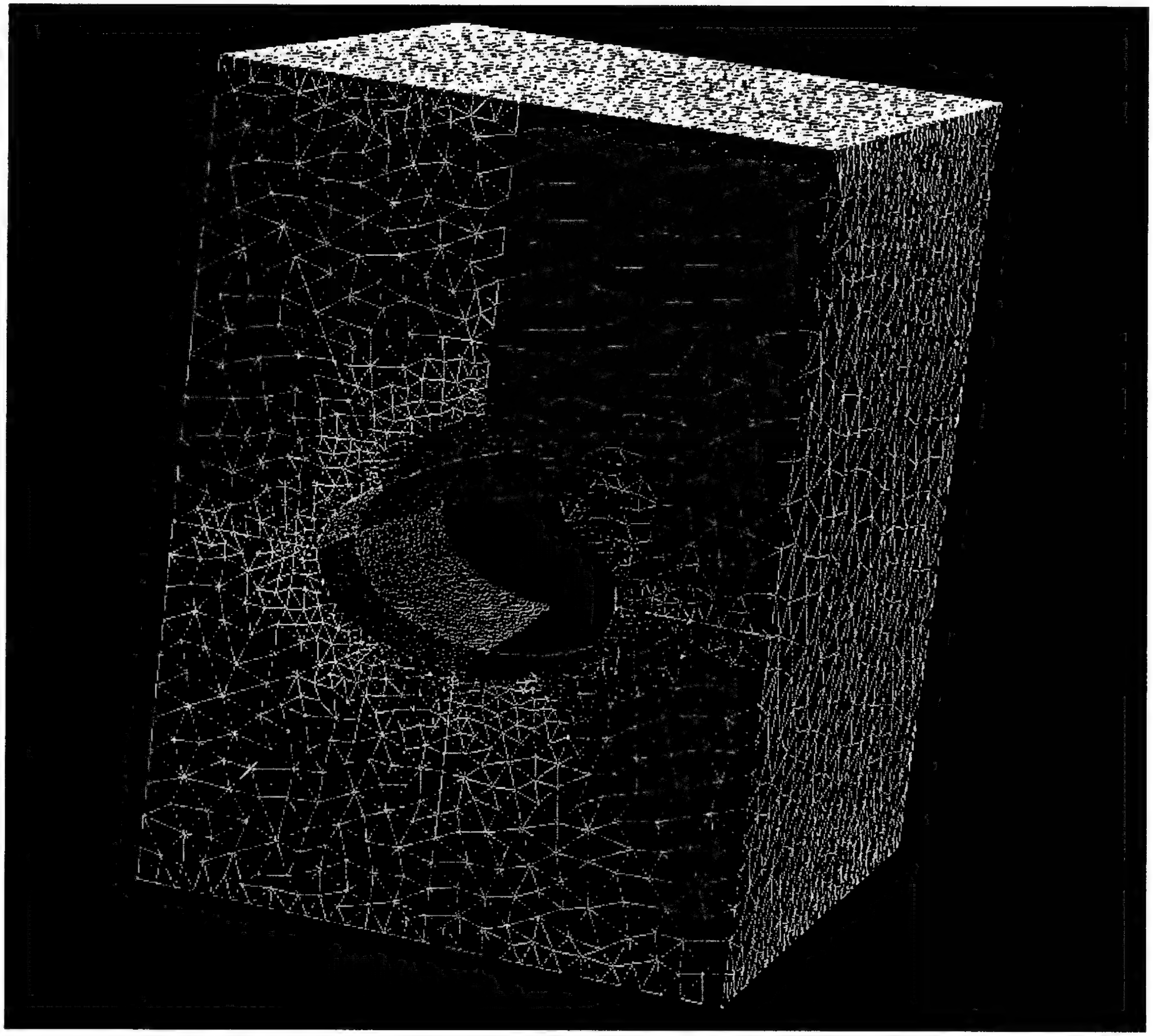
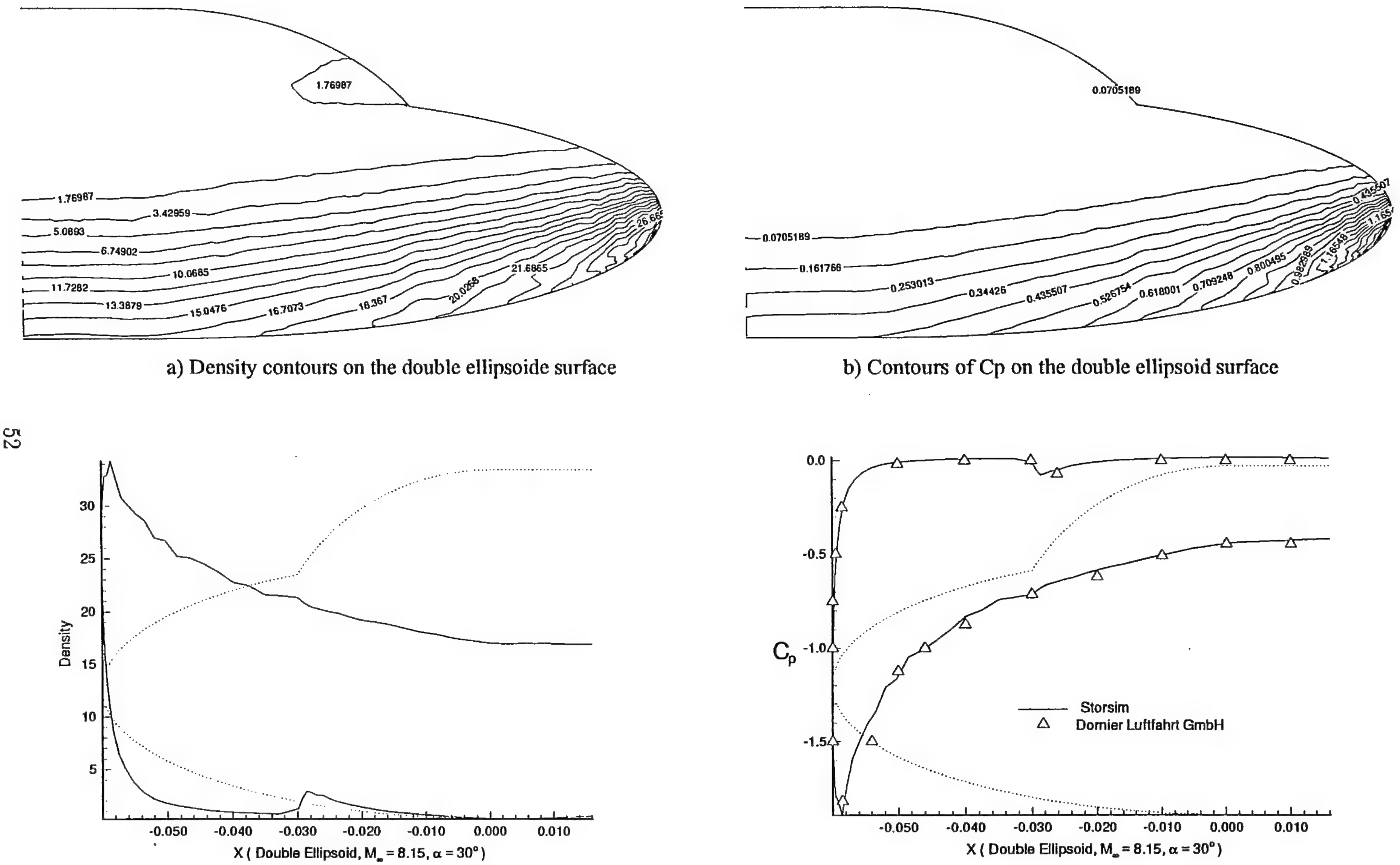


Figure 7.11: Another view of surface triangles. This shows the triangulation on part of the far–field boundary.



a) Density contours on the double ellipsoide surface

b) Contours of Cp on the double ellipsoid surface

c) Variation of density along the surface at midplane.

d) Variation of Cp at midplane

Figure 7.12: Hypersonic flow over a double ellipsoid.

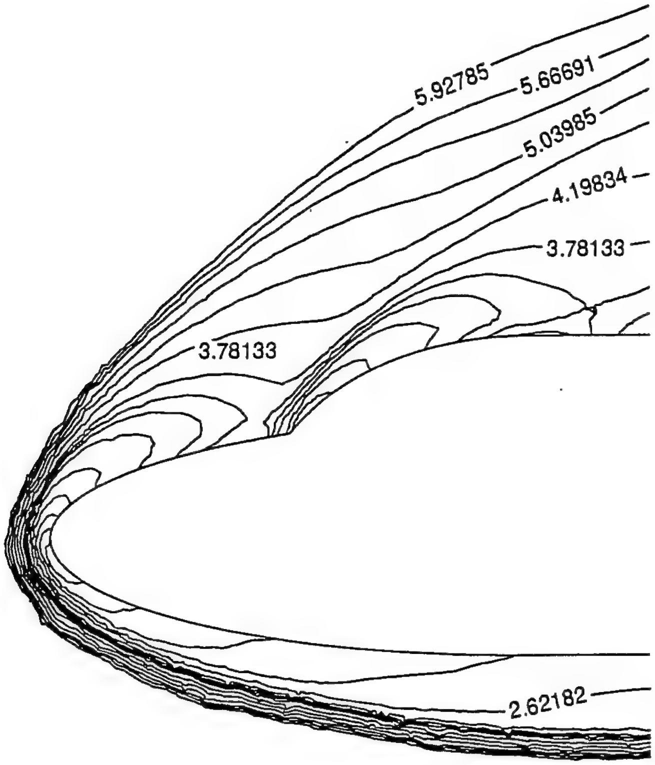


Figure 7.13: Hypersonic flow over double ellipsoid. Mach contours at the midplane

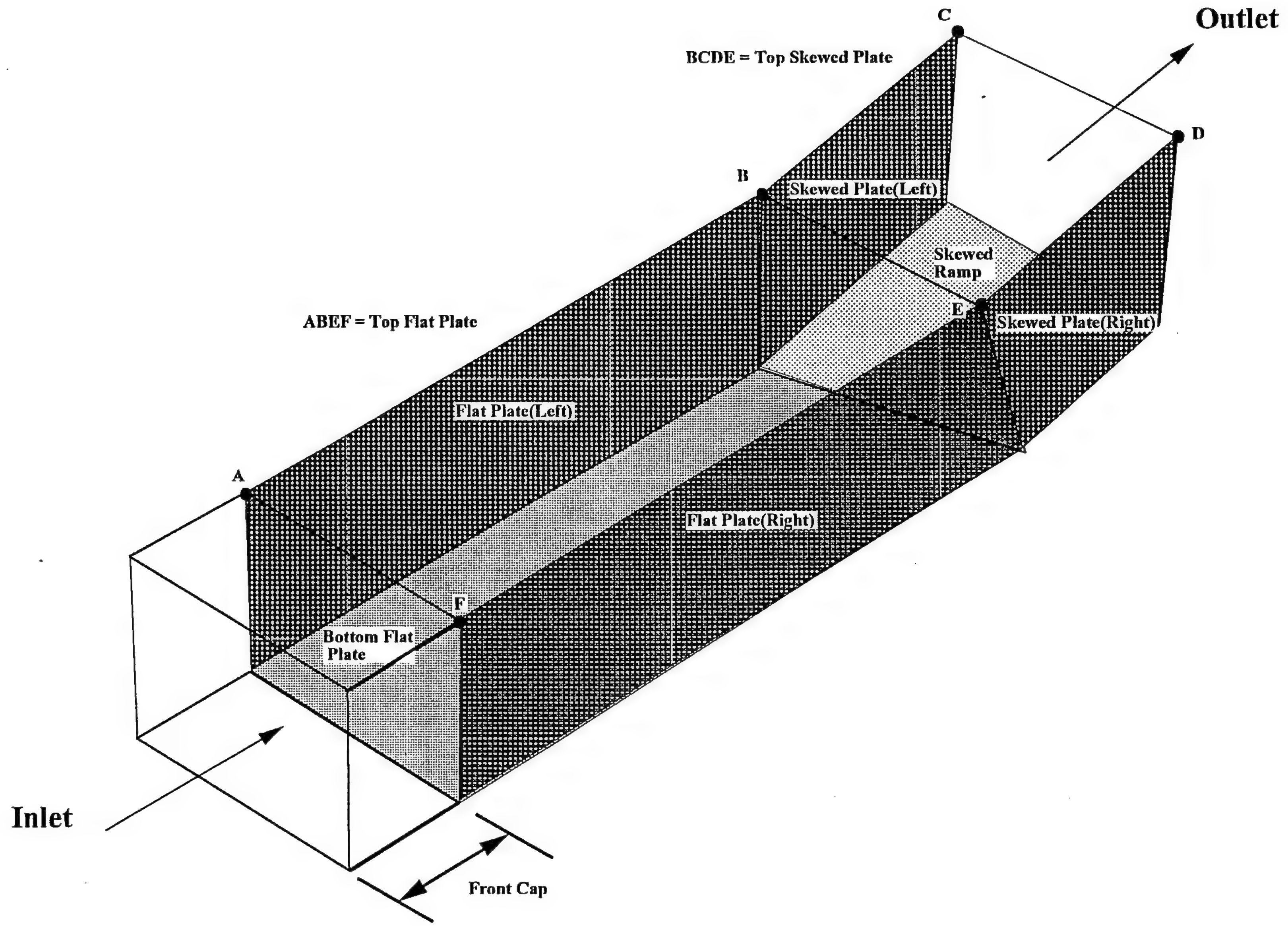


Figure 7.14: Schematic diagram of three−dimensional obstacle.

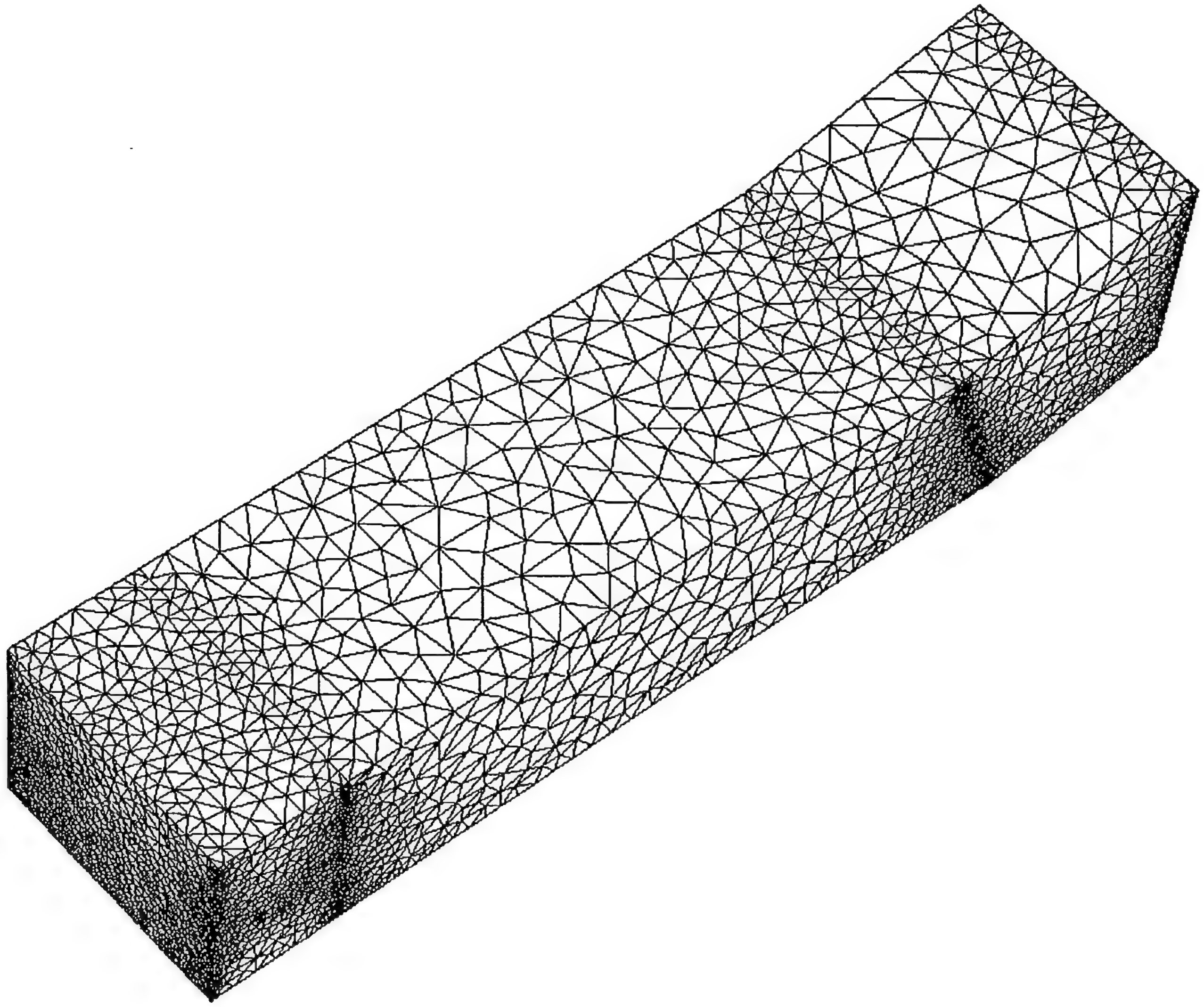


Figure 7.15: Surface triangulation on the boundary of the domain for the three–dimensional compression corner flow problem.

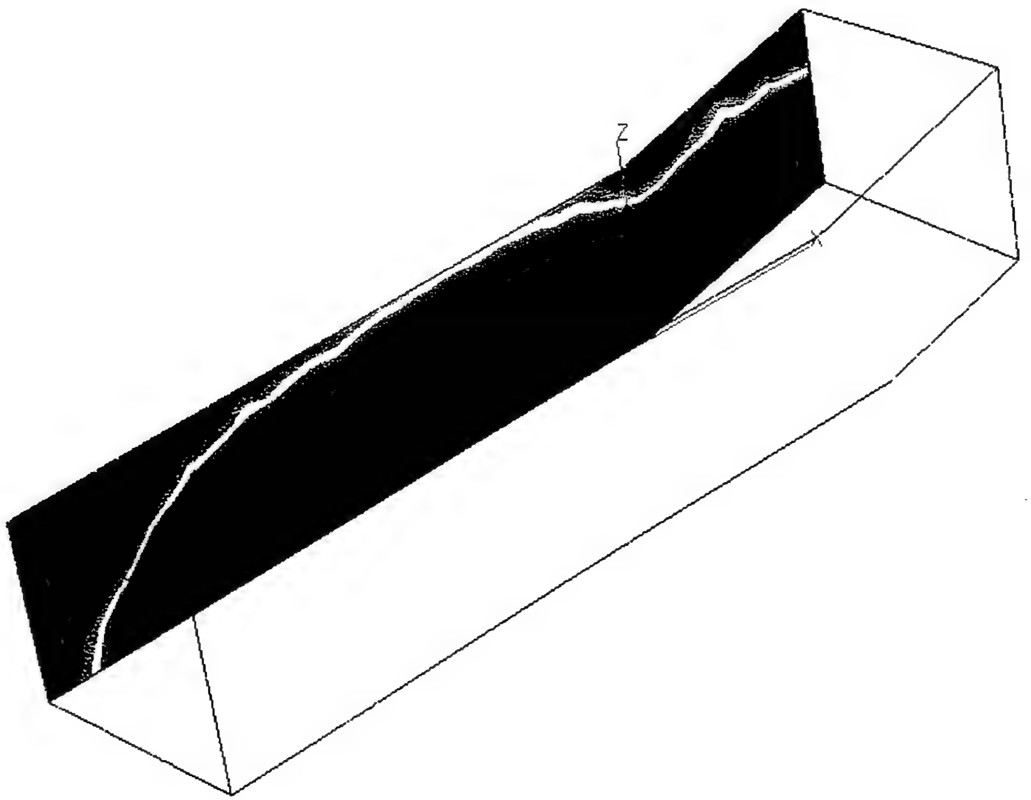


Figure 7.16: Mach number contours on the symmetry plane of flow over 3D obstacle.

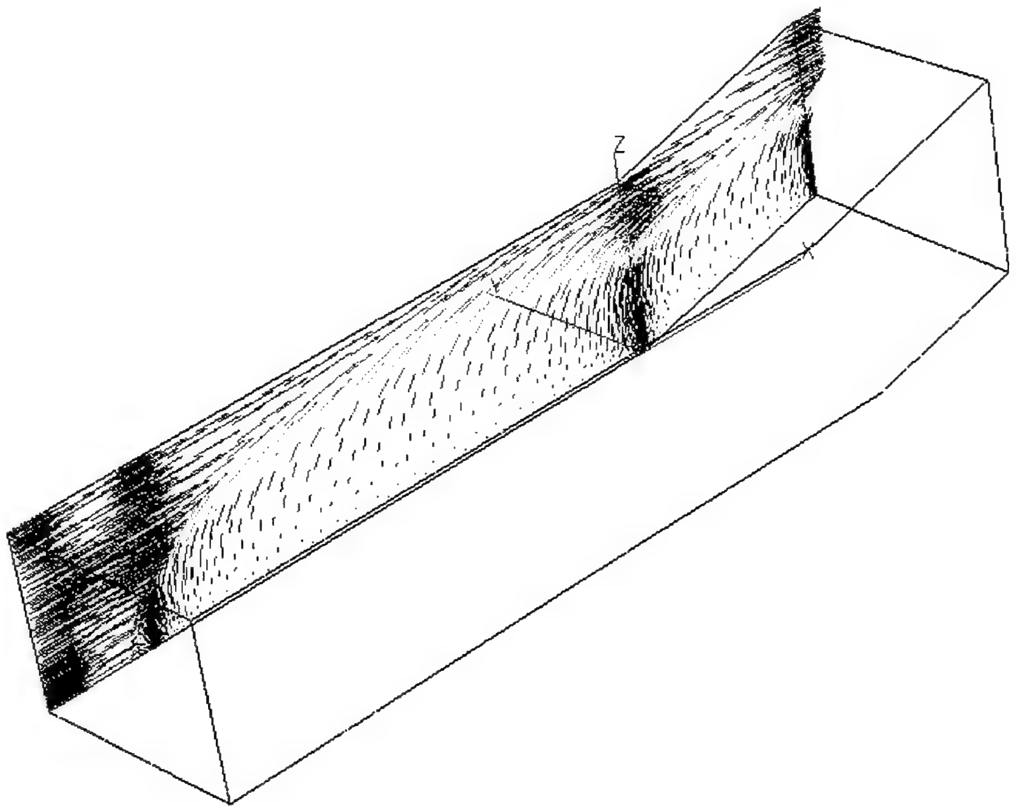


Figure 7.17: Velocity vectors on the symmetry plane of flow over 3D obstacle.

7.2.5 Flow of Real Gas over Cylinder

To demonstrate the real gas modeling capabilities of STOREFLOW, we consider hypersonic flow over a cylinder. The real gas simulation modeling in STOREFLOW, instead of using tables or correlations as used by many researchers, employs exact thermodynamical relations. The real gas model assumes equilibrium chemistry and a five species model for air modeled by statistical thermodynamics [1]. These five species are: molecular nitrogen molecular oxygen, nitric oxide, atomic oxygen and atomic nitrogen.

The free stream Mach number of the fluid is 10. This is a two-dimensional test case and the finite element mesh shown in Figure 7.18 consists of 2940 quadratic elements with proper biasing close to the cylinder. In Figure 7.19 we compare temperature and Mach contours for real gas flow analysis with those of perfect gas flow at the same Mach number. The temperatures at the cylinder wall and elsewhere in the domain are higher in the case of perfect gas simulation. Also, the real gas case shows a thinner shock layer and is closer to the cylinder. Contours the five species are shown plotted in Figure 7.20.

7.2.6 Generic Submunitions on CBU

In this case we generate tetrahedral mesh for a Cluster Bomb Unit (CBU) with four submunitions. The finite element mesh consists of 18 128 surface triangles on the boundaries. The fins of the CBU are modeled as *zero-thickness* surfaces. For such surfaces, the mesh generator automatically creates matching surface, domain boundary and corresponding surface triangles. Figure 7.21 shows surface triangulation of the CBU. After tetrahedralization, the mesh has 25 795 nodes and 136 474 tets. The complete mesh generation process required 4200 seconds on SGI Indigo r4000 running IRIX 5.3. The difference in the CPU time from previous problem is due to the complexity of the surface geometry in the problem.

Here we solved the problems using the Euler module of the flow solver. The freestream Mach number is 0.85. Contours of density, pressure, and Mach number on the surface of CBU are shown in Figure 7.22. Velocity vectors on the midplane are shown in Figure 7.23. These show recirculation in the cavities.

7.2.7 F-16

The F16 geometry was prepared by WL personnel and delivered to COMCO (Figure 7.24). This geometry consists of a detailed representation of the F16 with a fuel tank and a store mounted under each wing. A full set of experimental data has been obtained for the store for various free stream conditions [12]. The geometry was discretized using TETMESH. The mesh consists of 28 661 surface triangles, 119 829 nodes, 719 881 tetrahedral elements, and 95 300 prismatic elements surrounding the F16, fuel tank, and the store. CPU time required for mesh generation was 8500 seconds on an SGI Indigo r8000 running IRIX 6.0. Figure 7.25 shows triangular mesh on the F16 surface.

Figures 7.26 show contours of pressure coefficient C_p on F-16 and store for an inviscid run. Here the free-stream Mach number is 0.95 and the angle of attack is zero degrees. The solution

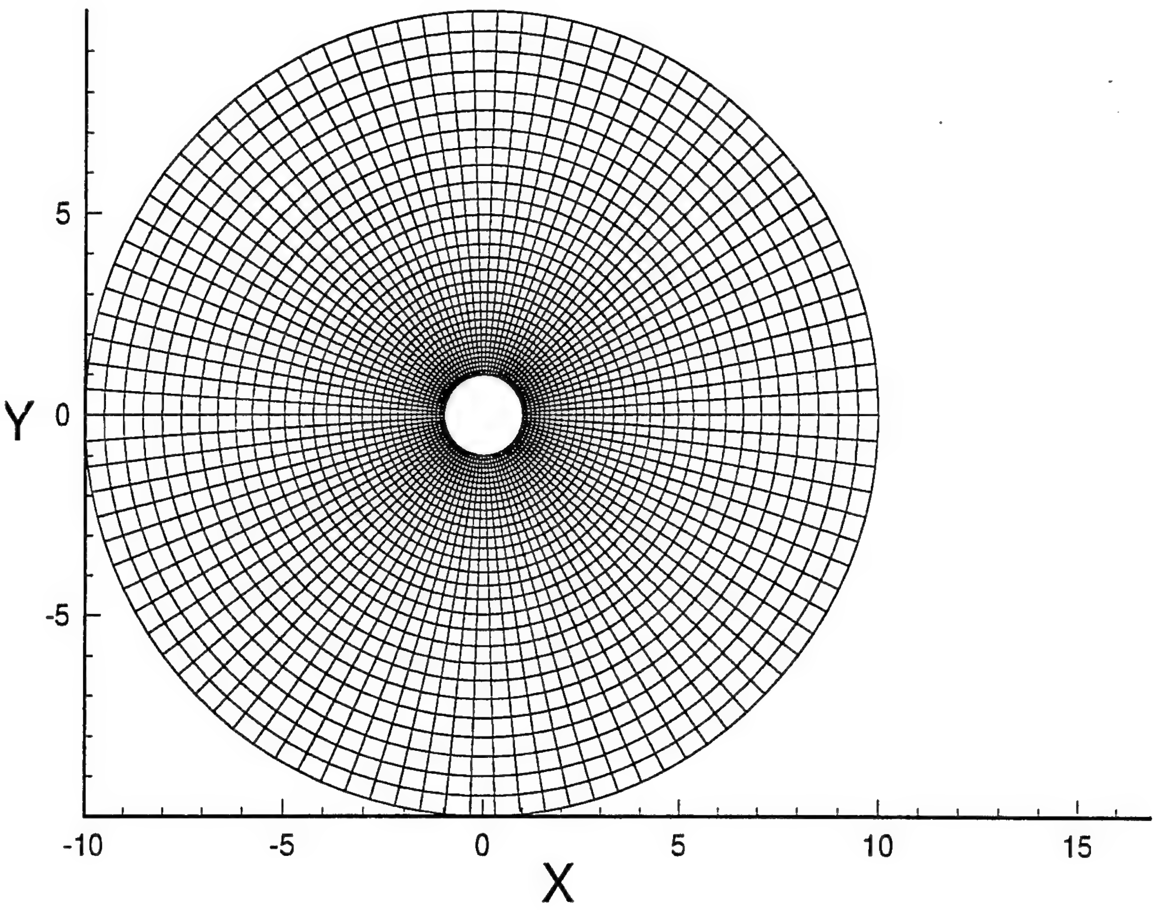
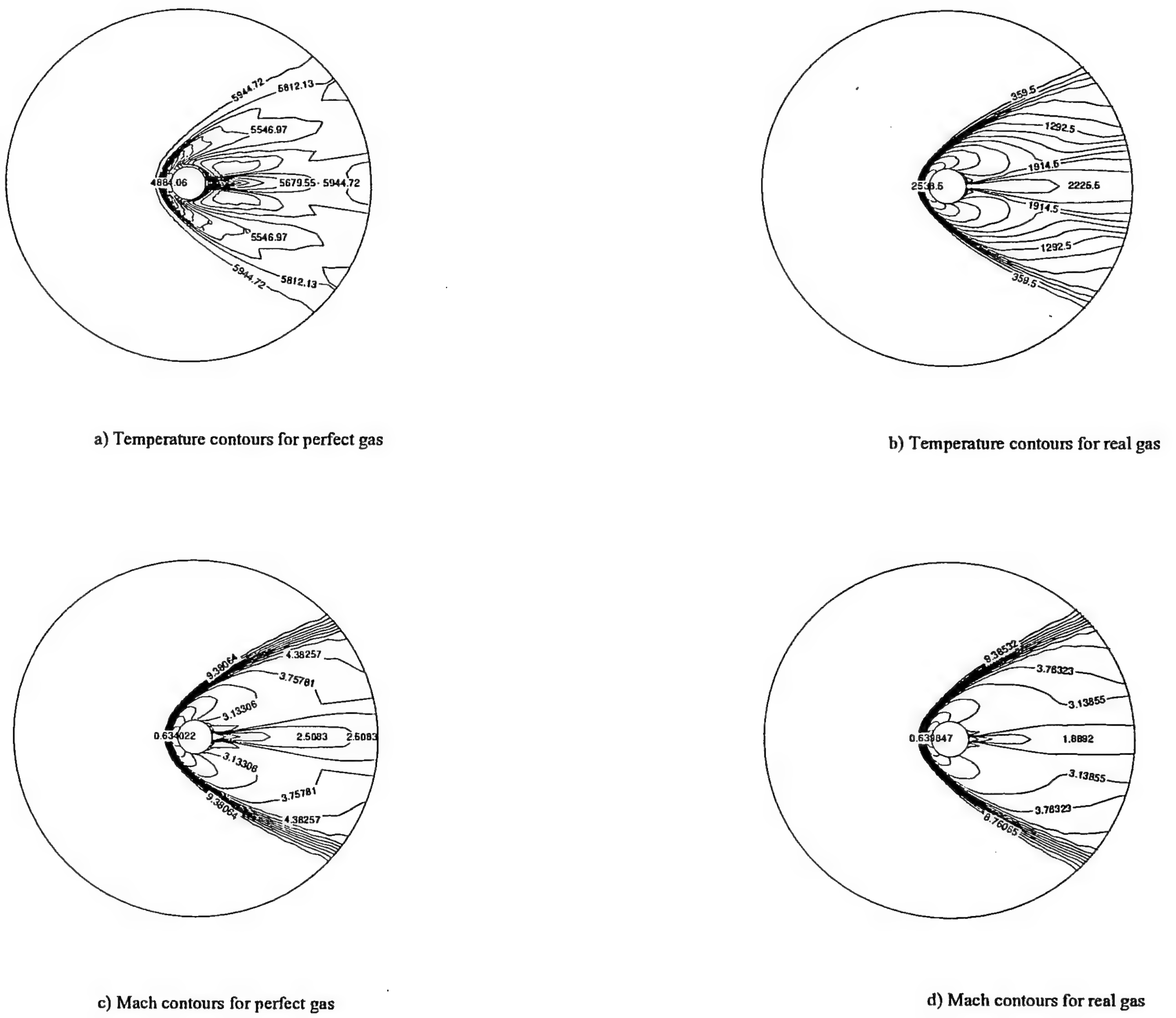


Figure 7.18 Finite element mesh for flow over a cyliner (M = 10)



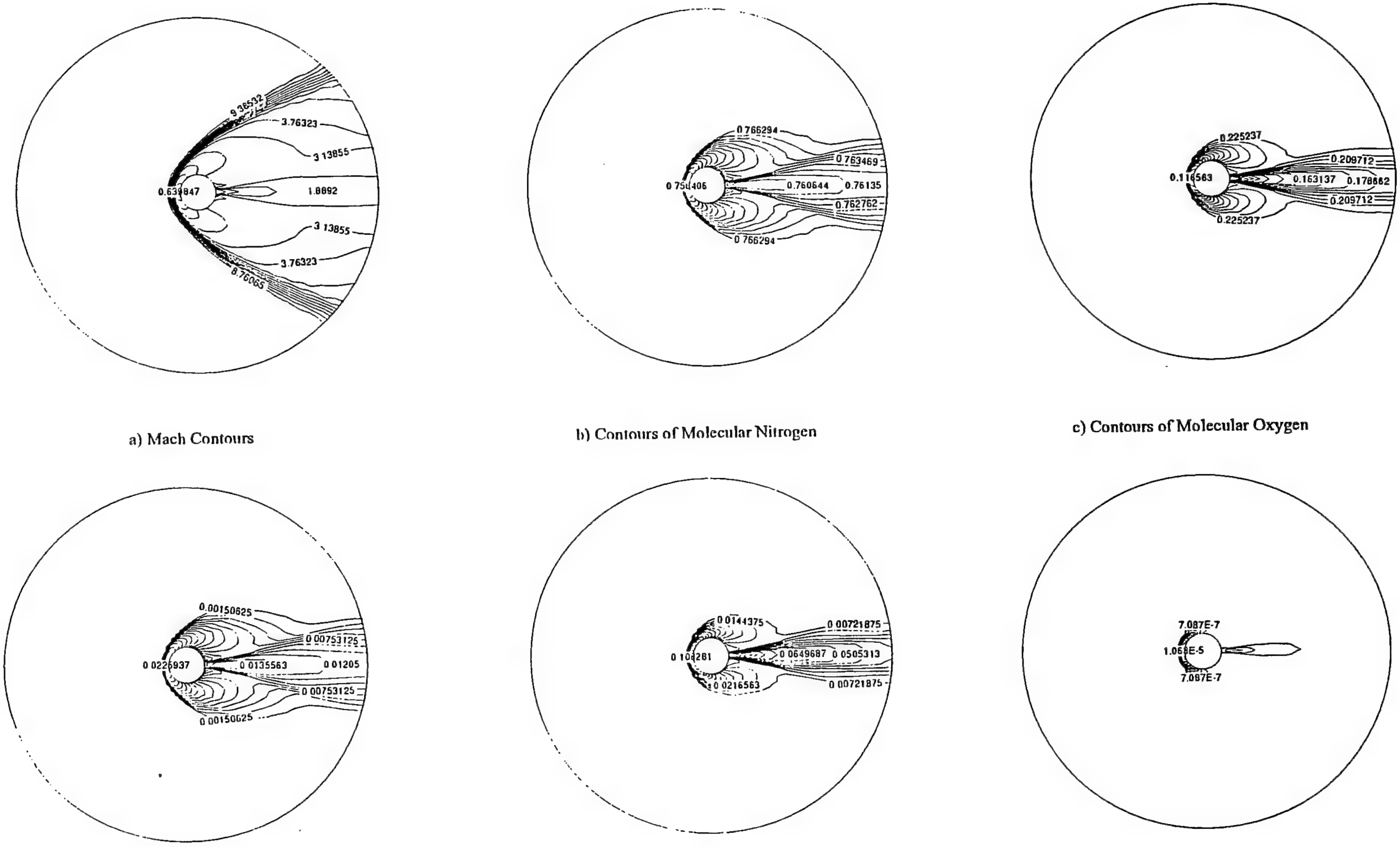
a) Temperature contours for perfect gas

b) Temperature contours for real gas

c) Mach contours for perfect gas

d) Mach contours for real gas

Figure 7.19: Comparison of temperature and Mach contours for real gas and perfect gas M = 10



a) Mach Contours

b) Contours of Molecular Nitrogen

c) Contours of Molecular Oxygen

d) Contours of Nitirc Oxide

e) Contours of Atomic Oxygen

f) Contours of Atomic Nitrogen

Figure 7.20: Real gas flow over a cylinder (M = 10). Contours of Mach number and five species of air

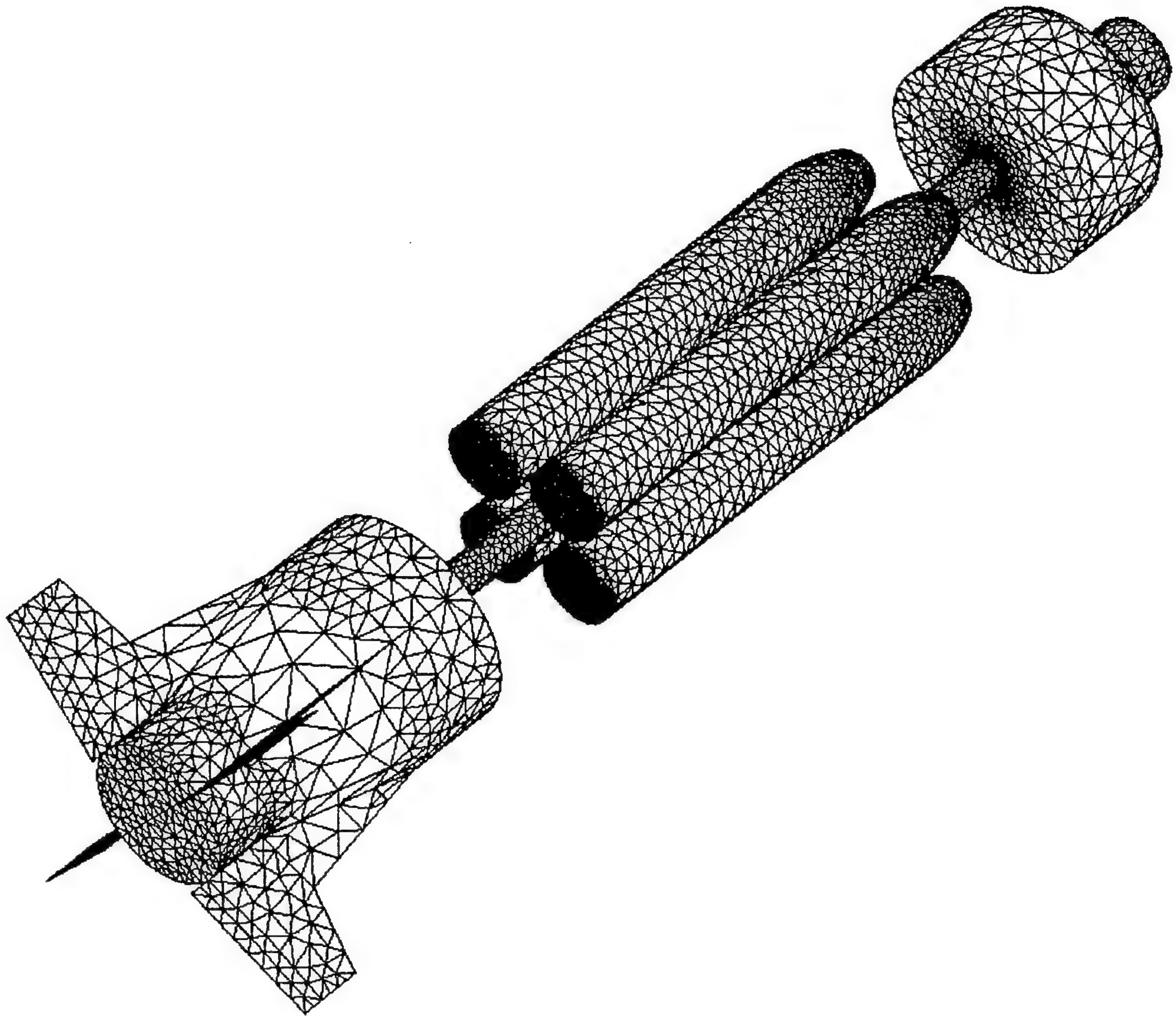
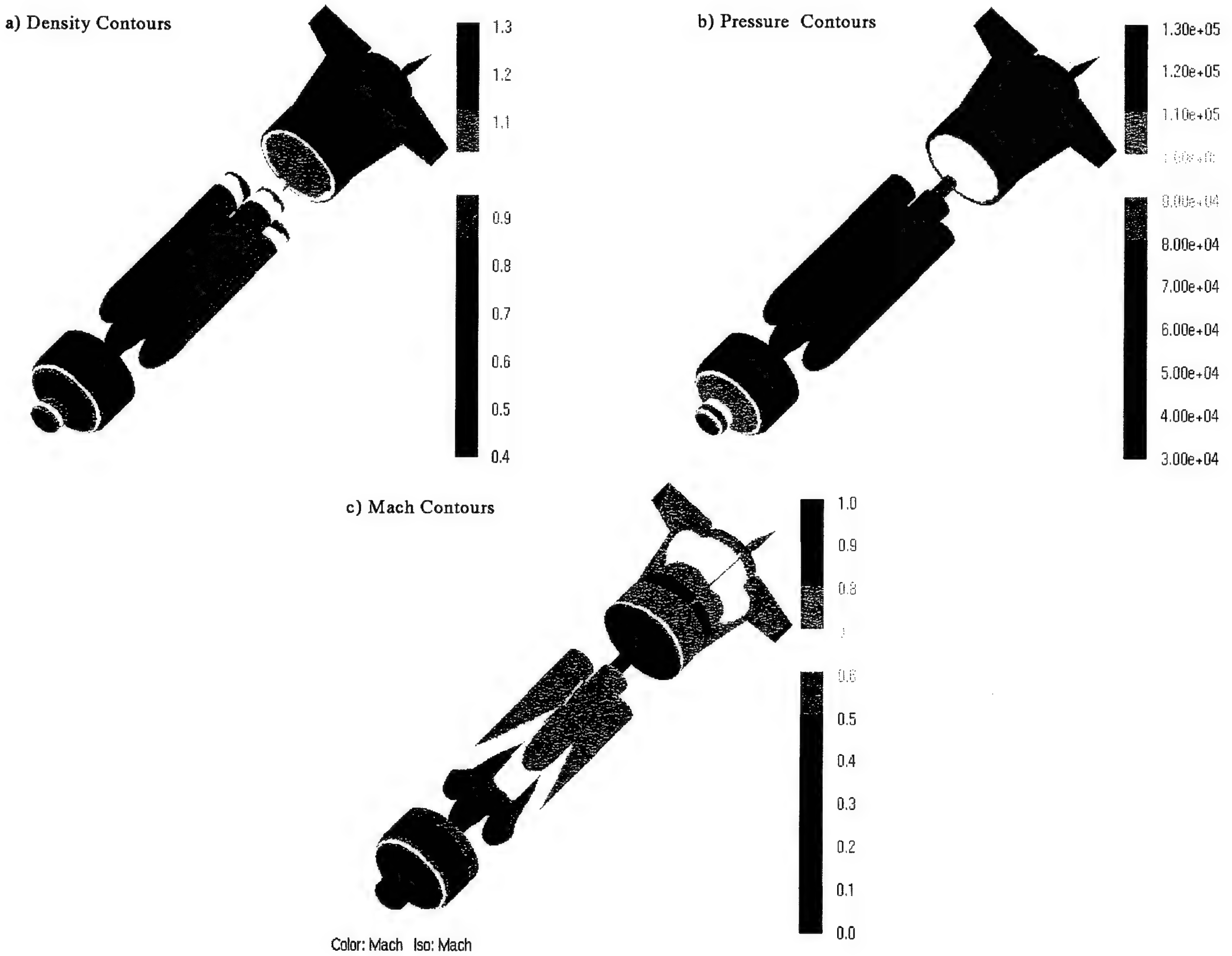


Figure 7.21: Surface triangulation of CBU geometry with 4 generic submunitions. Fins of CBU are defined as zero–thickness surfaces.



a) Density Contours

b) Pressure Contours

c) Mach Contours

Figure 7.22: Contours of density, pressure and Mach number on CBU with 4 submunitions.

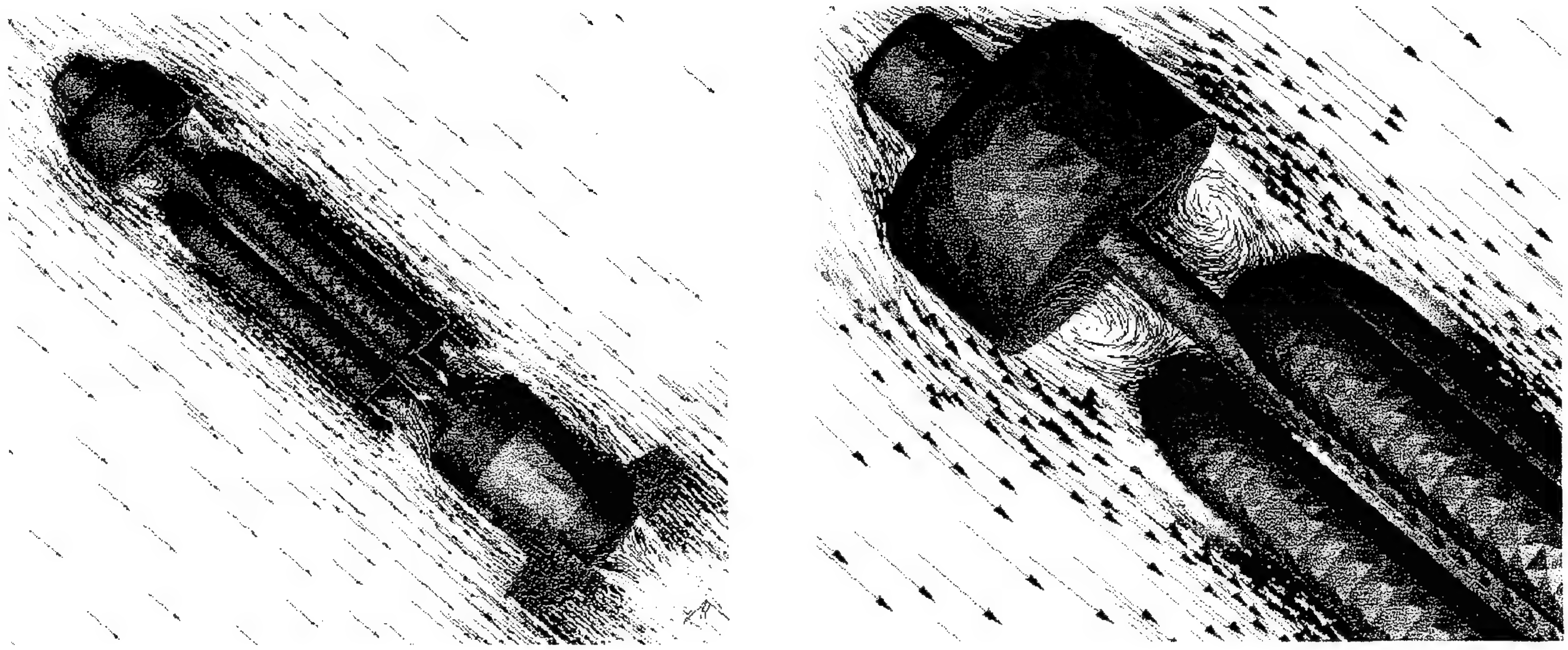


Figure 7.23: Velocity vectors on the midplane of the geometry showing recirculation regions.

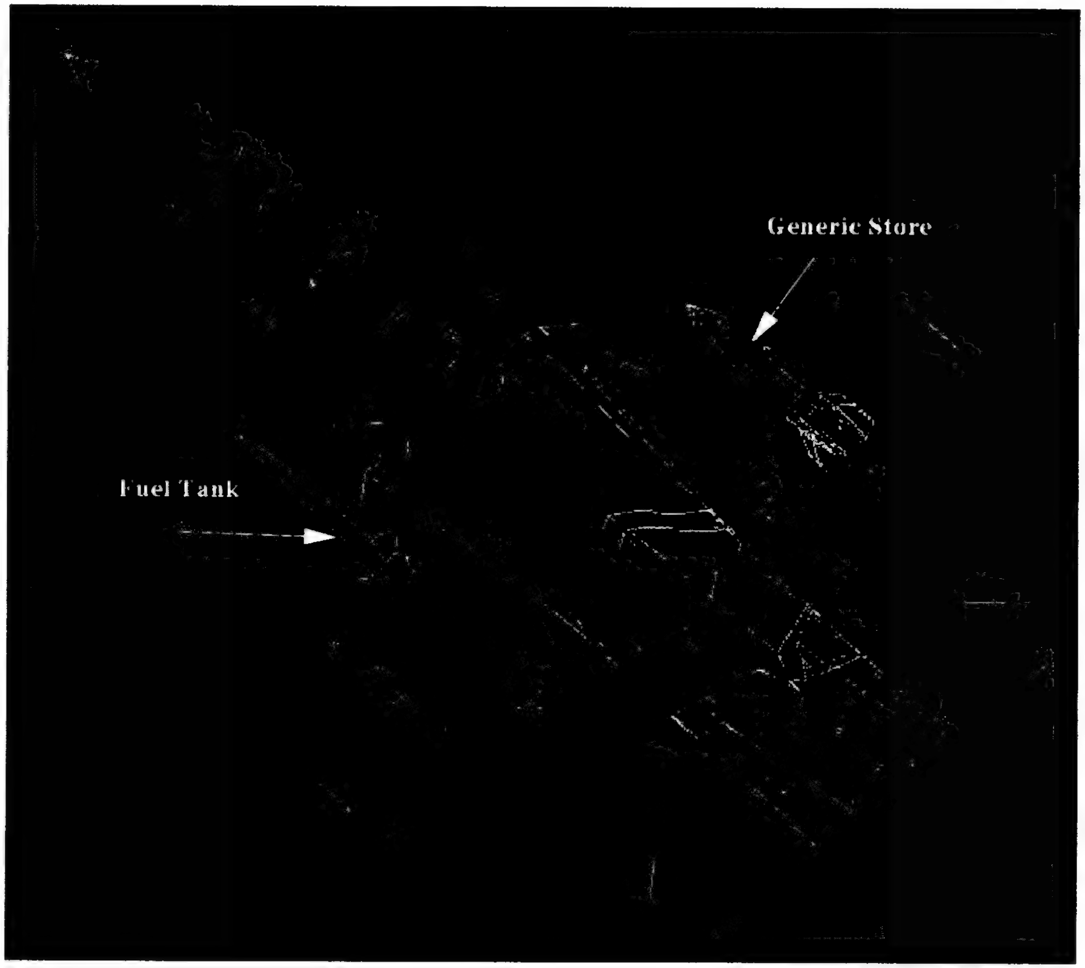


Figure 7.24: Surface geometry of F–16 with fuel tanks and generic store.

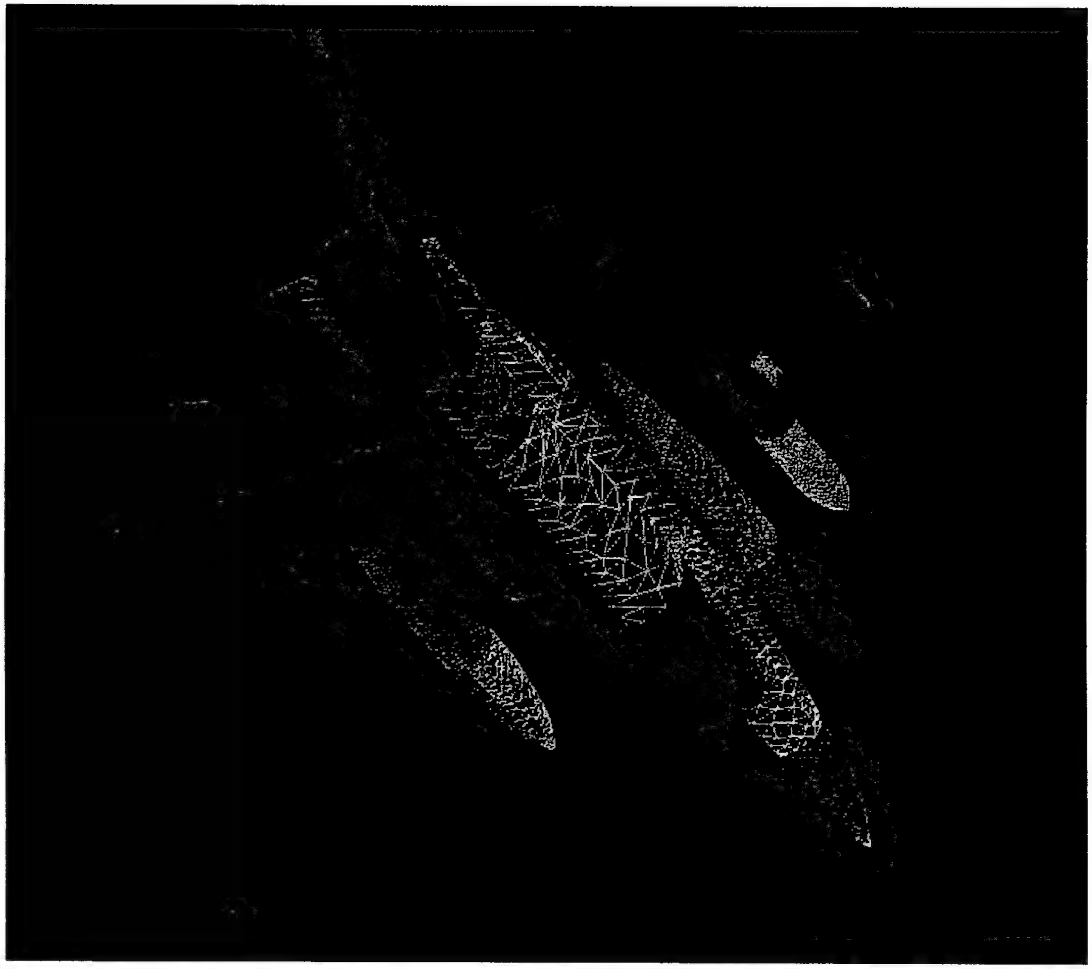


Figure 7.25: Surface triangulation on F–16 with fuel tanks and generic store.

was said to be converged when the non-dimensional solution norm was reduce by 4 orders of magnitude. It took 6500 time steps for to obtain a converged solution. Figure 7.27 shows the comparison pressure coefficient on the aircraft body at two different free-stream Mach numbers 0.95 and 1.1. In Figure 7.28 we compare the C_p on the fins of the store at $M = 0.95$ and 1.1. The computed results predict the overall trend and capture the flow behavior. The differences are attributed to the fact that the numberical solutions were obtained using inviscid flow assumption.

7.3 Demonstration Cases

7.3.1 F-16 with free falling generic store

Starting with the converged solution from previous inviscid F-16 run as the intial flow field, we released the store. The store separation calculations were performed using the integrated flow-, six-dof-, and mesh-moving, algorithm implemented in STORESIM. The analysis was run for another 3250 time steps. For each time step the location of the store was updated along with any required restructuring. The initial conditions of the store at the start of the separation are:

```
Step: 0 |DU.DV|/dt =         9.42779
[get_sixdof_motion]:: motion information at time: 0
com_velocity_tilde: 0 0 -0.203
       omega_tilde: 0 0 0
       quaternions: 0 0 0 1
    center_of_mass: 339.328 120 63.3017
```

The location of the center of mass and its velocity at different time steps is given in below. The data files and the solution save files are provided in the accompanying tape.

```
Step: 100 |DU.DV|/dt =         4.28933
[get_sixdof_motion]:: motion information at time: 0.00642502
com_velocity_tilde: 0.00283813 -0.00048171 -0.266419
       omega_tilde: 7.84291e-05 -7.99302e-05 -5.34408e-05
       quaternions: 1.25381e-07 -1.27756e-07 -8.53401e-08 1
    center_of_mass: 339.328 120 63.3002
           xode(1): 0.00642502
         xode(2:4): 0.00283813 -0.00048171 -0.266419
         xode(5:7): 7.84291e-05 -7.99302e-05 -5.34408e-05
        xode(8:11): 1.25394e-07 -1.27769e-07 -8.53487e-08 1
       xode(12:14): 339.328 120 63.3002

Step: 500 |DU.DV|/dt =         0.28565
[get_sixdof_motion]:: motion information at time: 0.0319988
com_velocity_tilde: 0.0141818 -0.00259118 -0.518533
```

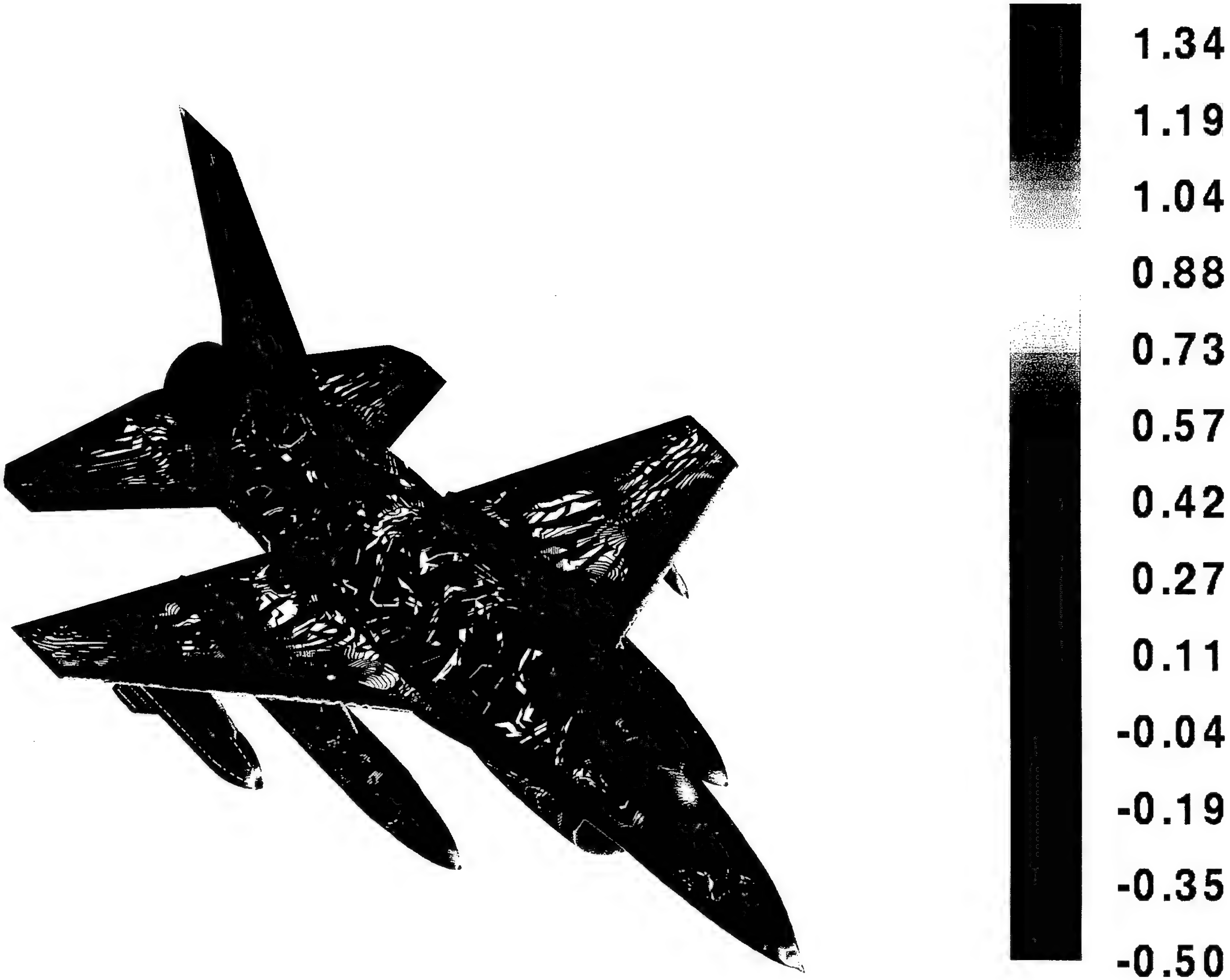
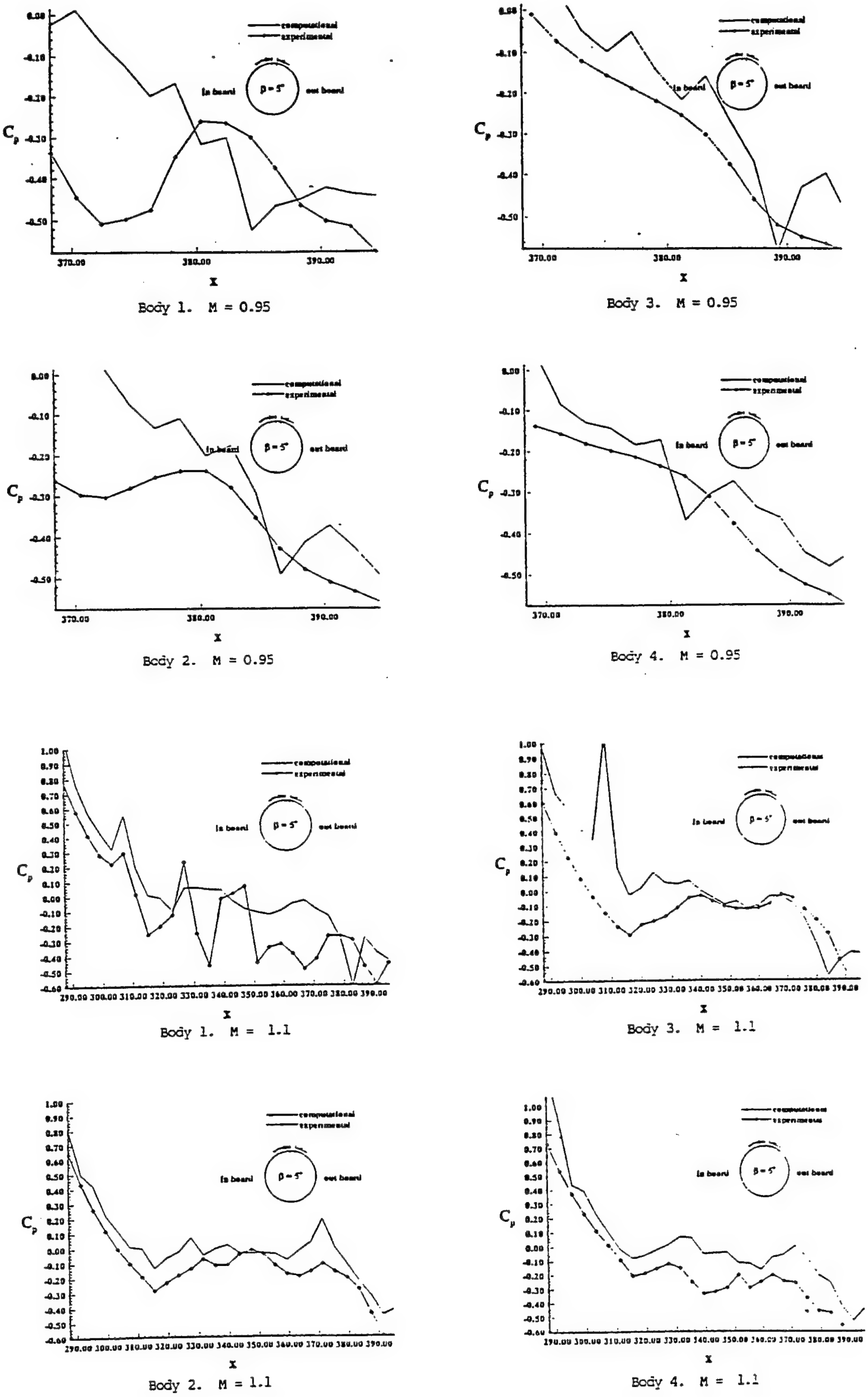


Figure ..26: F16$_p$ŒContour.



Body 1. M = 0.95

Body 3. M = 0.95

Body 2. M = 0.95

Body 4. M = 0.95

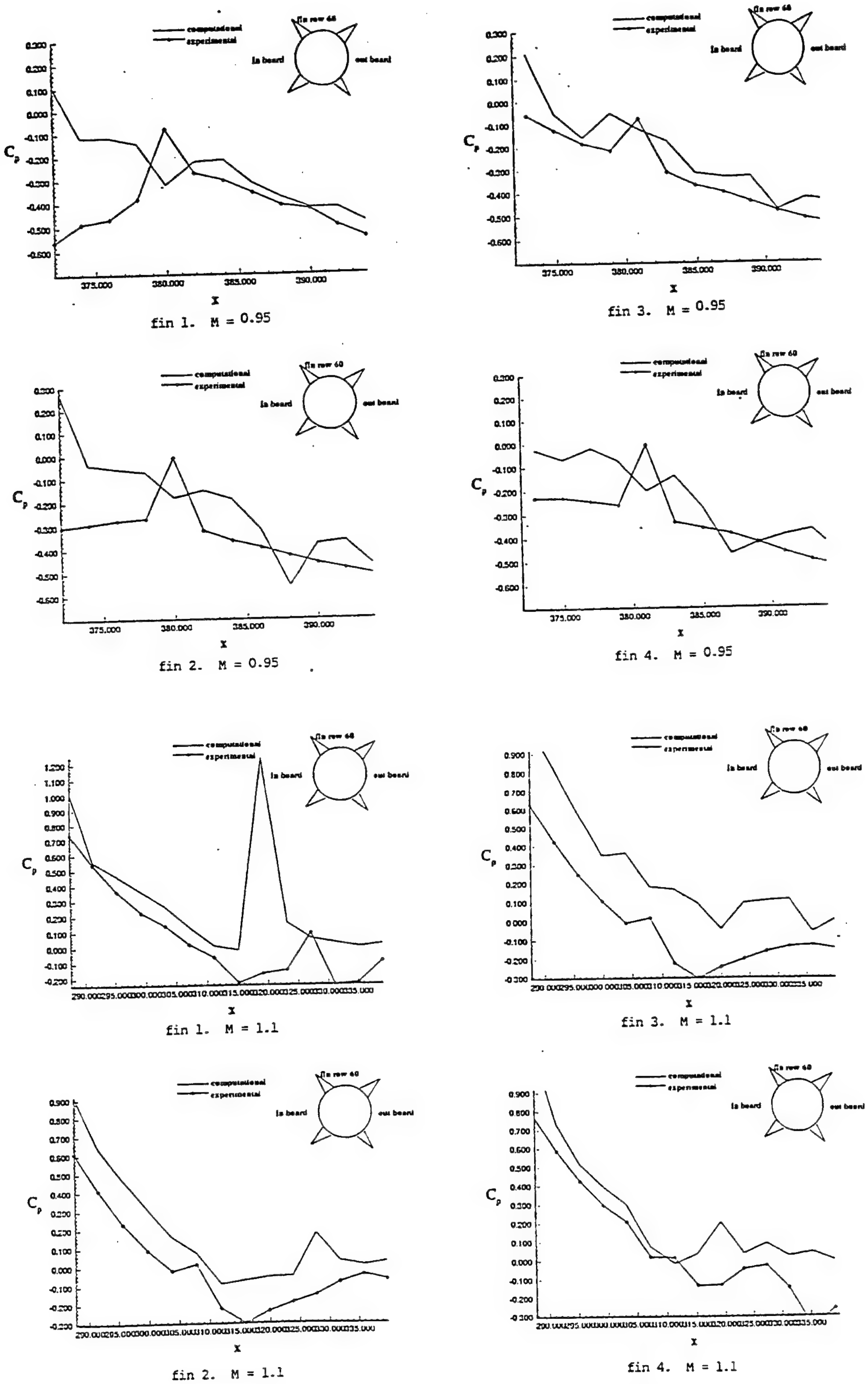
Body 1. M = 1.1

Body 3. M = 1.1

Body 2. M = 1.1

Body 4. M = 1.1

Figure 7.27: Comparison of computed values of Cp over F-16 with experimental data at M = 0.95 and 1.1



fin 1. M = 0.95

fin 3. M = 0.95

fin 2. M = 0.95

fin 4. M = 0.95

fin 1. M = 1.1

fin 3. M = 1.1

fin 2. M = 1.1

fin 4. M = 1.1

Figure 7.28: Comparison at computed values of Cp on the fins of store at M = 0.95 and 1.1

```
       omega_tilde: 0.000394509 -0.000402629 -0.000271088
       quaternions: 3.14511e-06 -3.20707e-06 -2.15391e-06 1
    center_of_mass: 339.328 120 63.2901
           xode(1): 0.0319987
         xode(2:4): 0.0141818 -0.00259118 -0.518533
         xode(5:7): 0.000394509 -0.000402629 -0.000271088
        xode(8:11): 3.14512e-06 -3.20709e-06 -2.15392e-06 1
       xode(12:14): 339.328 120 63.2901

Step: 1000 |DU.DV|/dt =        0.194849
[get_sixdof_motion]:: motion information at time: 0.063968
com_velocity_tilde: 0.0283269 -0.00559457 -0.833152
       omega_tilde: 0.000792428 -0.000813205 -0.000550542
       quaternions: 1.26306e-05 -1.2915e-05 -8.71066e-06 1
    center_of_mass: 339.328 120 63.2685
           xode(1): 0.063968
         xode(2:4): 0.0283269 -0.00559457 -0.833152
         xode(5:7): 0.000792428 -0.000813205 -0.000550542
        xode(8:11): 1.26306e-05 -1.2915e-05 -8.71067e-06 1
       xode(12:14): 339.328 120 63.2685

Step: 1500 |DU.DV|/dt =        0.154484
[get_sixdof_motion]:: motion information at time: 0.0959398
com_velocity_tilde: 0.0424375 -0.00895945 -1.14731
       omega_tilde: 0.001189 -0.00122945 -0.00083683
       quaternions: 2.84727e-05 -2.92358e-05 -1.9792e-05 1
    center_of_mass: 339.328 120 63.2369
           xode(1): 0.0959398
         xode(2:4): 0.0424375 -0.00895945 -1.14731
         xode(5:7): 0.001189 -0.00122945 -0.00083683
        xode(8:11): 2.84727e-05 -2.92358e-05 -1.9792e-05 1
       xode(12:14): 339.328 120 63.2369

Step: 2000 |DU.DV|/dt =        0.127969
[get_sixdof_motion]:: motion information at time: 0.127914
com_velocity_tilde: 0.0565076 -0.0126205 -1.4611
       omega_tilde: 0.00158002 -0.00164992 -0.00112796
       quaternions: 5.06172e-05 -5.2247e-05 -3.54926e-05 1
    center_of_mass: 339.328 120 63.1952
           xode(1): 0.127914
         xode(2:4): 0.0565076 -0.0126205 -1.4611
         xode(5:7): 0.00158002 -0.00164992 -0.00112796
        xode(8:11): 5.06172e-05 -5.2247e-05 -3.54926e-05 1
       xode(12:14): 339.328 120 63.1952
```

```
Step: 2500 |DU.DV|/dt =       0.106743
[get_sixdof_motion]:: motion information at time: 0.15989
com_velocity_tilde: 0.0705241 -0.0165259 -1.77459
       omega_tilde: 0.00196262 -0.00207458 -0.00142251
       quaternions: 7.89492e-05 -8.20148e-05 -5.58772e-05 1
    center_of_mass: 339.328 120 63.1434
           xode(1): 0.159892
         xode(2:4): 0.0705241 -0.0165259 -1.77459
         xode(5:7): 0.00196262 -0.00207458 -0.00142251
        xode(8:11): 7.89492e-05 -8.20148e-05 -5.58772e-05 1
       xode(12:14): 339.328 120 63.1434

Step: 3000 |DU.DV|/dt =      0.0918054
[get_sixdof_motion]:: motion information at time: 0.191867
com_velocity_tilde: 0.0844736 -0.020651 -2.08782
       omega_tilde: 0.00233644 -0.0025036 -0.00172001
       quaternions: 0.000113327 -0.000118608 -8.0995e-05 1
    center_of_mass: 339.328 120 63.0817
           xode(1): 0.19187
         xode(2:4): 0.0844736 -0.020651 -2.08782
         xode(5:7): 0.00233644 -0.0025036 -0.00172001
        xode(8:11): 0.000113327 -0.000118608 -8.0995e-05 1
       xode(12:14): 339.328 120 63.0817

Step: 3250 |DU.DV|/dt =       0.086224
[get_sixdof_motion]:: motion information at time: 0.207855
com_velocity_tilde: 0.0914193 -0.0227924 -2.24434
       omega_tilde: 0.00252038 -0.00271971 -0.0018699
       quaternions: 0.000132742 -0.000139485 -9.53437e-05 1
    center_of_mass: 339.328 120 63.0471
           xode(1): 0.207859
         xode(2:4): 0.0914193 -0.0227924 -2.24434
         xode(5:7): 0.00252038 -0.00271971 -0.0018699
        xode(8:11): 0.000132742 -0.000139485 -9.53437e-05 1
       xode(12:14): 339.328 120 63.0471
```

The displacement of the store at the end of 3250 time steps, after converting the units to the scaled down model units, is the vertical displacement of the store is 11 inches.

in [26]

7.3.2 Hypersonic 2-Stage Case

Details of this demonstration case are presented in [6]

8 Conclusions

The primary goal of this effort was to couple these disciplines into a unified computational environment for analyzing dynamic separation of weapons, stages, and crew escape vehicles from supersonic and hypersonic parent vehicles. The major outcome of this research and development effort is a software package called STORESIM. The STORESIM package is a tightly integrated group of modules (TETMESH, STOREFLOW, STOREVIS) that performs grid generation, static flow simulation, six degree-of-freedom rigid body dynamics, dynamic flow analysis, body movement, grid-node movement, grid restructuring, and time-dependent results visualization.

Key accomplishments of this effort are:

1. *Mesh Generation:* A grid generator is essentially useless without a powerful technique for defining the geometry to be discretized. To this end we have developed an unstructured mesh generation module, TETMESH, that is capable of reading the geometry in a very simple network/trimmed surface format. Salient features of this module include:

    - *Triangulation:* Surface triangulation is based on the Dulanay criterion. The surface triangulator is capable of handling complex curved surfaces with cusped edges.
    - *Volume Gridding:* The volume mesher generates tetrahedral elements with an option of generating prismatic elements extruded from the surfaces. The tetrahedralization is also based on Dulanay criterion with proper face enforcement to fill the voids.
    - Zero-thickness Surface: Thin fins protruding from a missile can be approximated as single surfaces during data preparation. TETMESH automatically generates a matching surface to create a barrier and prevent fluid movement across this surface.
    - Adaptivity: The meshing module is capable of refining/unrefining the mesh. Regions of maximum solution gradients are automatically refined to better capture solution in critical regions.

2. *Flow Solver:* The STORESIM package has a compressible fluid flow solver module called STOREFLOW. Key features of the STOREFLOW module are:

    - An option for ALE (Arbitrary Lagrangian Eulerian) computations. This essentially makes it possible to use moving meshes to model multiple moving bodies. This new ALE formulation is extremely robust, and can even produce accurate results for bodies launched at high speed into ambient air.
    - A very sophisticated far stream boundary condition designed to prevent spurious energy build up and consequent reflections at transonic speeds. This approach also includes modifications to account for moving mesh boundaries.
    - Euler and Navier-Stokes versions. Turbulence modeling based on a simple zero equation model has been tested for two-dimensional flows. The extension to three-dimensional turbulence modeling, although not implements should be straightforward.

- Real gas capability, based on a 5 species (Nitrogen, Oxygen, Atomic Nitrogen, Atomic Oxygen and Nitric Oxide) model for dissociated air.
- Multiple element types (tetrahedra, Hexahedra or Prisms). Additional element types can be easily incorporated into the code.

3. *Six-DOF Solver:* STOREFLOW is fully integrated with a six degree-of-freedom (6 dof) solver for rigid body dynamics, and a meshmover, that moves the mesh in response to body motions.

4. Node Movement: An innovative grid movement/restructuring method based on the grid-node-flow analogy and generalized edge swapping in three-dimensions. The node movement/restructuring is required when modeling body movements in the flow field.

5. *Visualization:* The visualization package, STOREVIS, is capable of post-processing structured/unstructured grids and producing color plots of slices, isosurfaces, velocity vectors, and particle traces.

8.1 Future Work

STORESIM has been used to solve several complex problems. In every case, all relevant modules of the STORESIM package are used. During the development and final validation process, each module has been tested separately and as a complete package for accuracy and robustness. Based on these tests, we strongly feel that STORESIM is capable of generating unstructured meshes on complex geometries with very little user intervention. The flow solver and six-dof solvers are very stable and produce accurate results. Since this was a major development effort aimed at alleviating the problems with mesh generation and modeling multi-body CFD problems, entire effort was put in implementation of accurate and robust techniques. However, there are several areas, such as interfacing with third party flow solvers, implementation of better turbulence models, that need additional work.

9 Acknowledgments

This work was supported by the Air Force Wright Laboratory, Flight Dynamics Dictorate, Wright-Patterson AFB, contract F33615-90-C-3001, F. Witzeman and J. Nelson technical monitors.

10 References

[1] Anderson, Jr. J. D., *Hypersonic and High Temperature Gas Dynamics,* McGraw Hill, NY 1989.

[2] Barth, T. J., Wiltberger, N. L., and Gandhi, A. S., "Three-Dimensional Unstructured Grid Generation via Incremental Insertion and Local Optimization," NASA CP-3143, Software Systems for Surface Modeling and Grid Generation, April, 1992.

[3] Blake, K. R., and G. S. Spragle, "Unstructured 3D Delaunay Mesh Generation Applied to Planes, Trains, and Automobiles," AIAA 93-0673, Reno, 1993.

[4] Brown, P. N., and Saad, Y., "Hybrid Krylov Methods for Nonlinear Systems of Equations". SIAM J. SCI STAT COMPUT., VOL 11, No. 3, pp 450 -481, May 1990.

[5] Chalot, F., and Hughes, T. J. R., "A consistent equilibrium chemistry algorithm for hypersonic flows". Computer Meth. in App. Mech. and Engi. 112 (1994) 25 $\sim$ 40

[6] Crosby, W. A., "Separation Tests of the BETA Two-Stage- To-Orbit Vehicle Models at Mach 8," AEDC-TR-94-9, May 95.

[7] Dortmann, K., "Viscous and Inviscid Hypersonic Flow About a Double Ellipsoid", Workshop on Hypersonic Flows for Reentry Problems, Part II,pp. 15-36, Antibes, France, April 1991.

[8] Goswami, A., "Grid Restructuring for Moving Boundaries," Master's Thesis, Dept. Aerospace Engineering, University of Texas at Arlington, 1990.

[9] Hazlewood, C. T., "A Divide and Conquer Approach to d-Dimensional Triangulations," Ph.D. dissertation, University of Texas at Austin, 1988.

[10] Hazlewood, C. T., "Constrained Simplicial Tessellations in E^3," manuscript, University of Texas at Austin, 1989.

[11] Hazlewood, C. T., "Implementing an Algorithm for Three-Dimensional Triangulations that Include Arbitrary Triangular Regions," manuscript, University of Texas at Austin, 1989.

[12] Hopf. J. C., "Separation Characteristics of the DWS-24 Dispenser, the MK-84 LDGP, the 370-GAL Tank(E) + Pylon, and the Generic Missile (Metric and Pressure-Instrumented) in the Flow Field of the F-16 Aircraft." AEDC-TSR-94-P1, 1994.

[13] Hughes, T. J. R., Franca, L. P., and Mallet, A.,"A New Finite Element Formulation for Computational Fluid Dynamics (I).", CMAME, 1985.

[14] Hughes, T. J. R., et al., "A new finite element formation for computational fluid dynamics (Vols. I through V)", Computer Methods in Applied Mechanics and Engineering. (1985)

[15] B. Joe, "Construction of K-Dimensional Delaunay Triangulations using Local Transformations," SIAM J. Sci. Comput. 14, No. 6, pp. 1415-1436, Nov. 1993.

[16] Kennon, S. R., Meyering, J. M., Berry, C. W., and Oden, J. T., "Geometry Based Delaunay Tetrahedralization and Mesh Movement Strategies for Multi-Body CFD," AIAA paper No. 92-4575, Hilton Head, SC, 1992.

[17] Lawson, C. L., "Properties of n-dimensional Triangulations," CAGD 3 (1986), 231-246.

[18] Lou, Jean-Ching, "A New Class of Decomposition for Inverting Asymmetric and Indefinite Matrices." Comp. Meth. in Appl. Vol. 25, No. 4, pp ~ 104, 1993 .

[19] Ruppert, J., and R. Seidel, "Difficulty of Tetrahedralizing 3-Dimensional Non-Convex Polyhedra", *Proceedings of the Fifth Annual Symposium on Computational Geometry*, 1986, 380-392.

[20] Shakib, F., *Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations,*, Ph.D., Dissertation, Stanford University, CA, 1988.

[21] Tworzydlo, W. W., Oden, J. T., and Thronton, E. A., "Adaptive Implicit/Explicit Finite Element Method for Compressible Viscous Flows," Comput. Meth. Appl. Mech. Engng., Vol 95, pp. 397-440, 1992.

[22] Venkatasubban, C. S., "*Développment* d'une *méthod d'éléments* finis pour le calcul *d'éconlements* compressible tridimensionnels en *hypothése* fluid parfait. *Thése* de doctorat, Ecole Nationale *Supérienre* de l'A'eronautique et de l'Espace, Tonlonse, France (1991)

[23] Venkatasubban, C. S., "A new finite element formulation for ALE (Arbitrary Lagrangian Eulerian) compressible fluid mechanics", Int. Journal of Engineering Science, Vol. 33, No. 12, pp. 1743 ~ 1762, 1995

[24] Weatherill, N. P., and O. Hassan, "Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints," IJNME, 37, pp. 2005-2039 (1994).

[25] Wigton, L.B., "GMRES, Preconditioning, conjugate gradients and all that". Boeing Rept.

[26] "Demonstration/Verification of STORESIM," Final Report, Task 4, Aeromechanics Technology, Contract F33615-94-C-3009, Northrop Grumman Corp, Dec 96.

A Appendix - A

A.1 Flux Jacobian for the Navier - Stokes Equations

The Navier-Stokes equations expressed in terms of conservation variables are:

$$\boldsymbol{U}, t + F^e_{i,i} = F^d_{i,i} \tag{A.1}$$

F^e_i is the Euler Flux Vector. (A.1) can be re-written as:

$$u_{,t} + A_i U_{,i} = F^d_{i,i} \tag{A.2}$$

where $A_i = F^e_{i,\boldsymbol{U}} = EulerFluxJacobian.$ Formulation for A_i can be found in many standard references.

A.2 Flux Jacobians in Terms of Entropy Variables

The Entropy variables are defined as:

$$V^T = (\rho s)_{,\boldsymbol{U}} \tag{A.3}$$

where, ρ is the density of the fluid, and s is the entropy. Use of these variables symmetrizes the Navier-Stokes Equations:

(A.1) becomes

$$\tilde{A}_0 V_{,t} + \tilde{A}_i V_{,i} = \left[\tilde{K}_{ij} V_{ij}\right]_{,i} \tag{A.4}$$

where

$$\left.\begin{array}{l} \tilde{A}_0 = u_{,v} \\ \tilde{A}_i = F^e_{i,V} = F^e_{i,u} u_{,v} = A_i \tilde{A}_0 \\ F^d_i = \tilde{K}_{ij} V_{ij} \end{array}\right\} \tag{A.5}$$

The matrices $\tilde{A}_0, \tilde{A}_i, \tilde{K}_{ij}$ are defined in term of the entropy (or conservation) variable in [20].

A.3 Streamline Operator I_{SUPG}

$$I_{SUPG} = \int_\Omega (\tilde{A}_i^{ALE}\ W_{,i}) \ \cdot \ [\tau](\tilde{A}_i^{ALE}\ V_{,i}) d\Omega \tag{A.6}$$

$$\begin{array}{llll} & \tilde{A}_i^{ALE} & = (A_i - u_i^g\ I)\tilde{A}_o^{ALE} & = A_i^{ALE} \cdot \tilde{A}_o \\ \text{where} & V & = \text{Vector of entropy variables} & = (-\rho\ s)_{\boldsymbol{U}} \\ & \tilde{A}_o & = \boldsymbol{U}_{,V} & \\ & [\tau] & = \frac{h}{2}\tilde{A}_o^{-1}|A_i^{ALE} A_i^{ALE}|^{-1/2} & \end{array}$$

A.4 Shock Operator I_{shock}

$$I_{shock} = \int_\Omega W \cdot (\nu_d - \ \neq_t au)\tilde{A}_o V_{,i} d\Omega \tag{A.7}$$

where,

$$\begin{aligned} \nu_d &= \left[\frac{(\tilde{A}_i^{ALE}\ V_{,i}) \cdot \tilde{A}_o^{-1}(\tilde{A}_j^{ALE}\ V_{,i})}{(\xi_{k,m} V_{,m}) \cdot (\tilde{A}_o \xi_{k,n} V_{,n})}\right]^{1/2} = \text{scalar} \\ \nu_\tau &= \left[\frac{(\tilde{A}_i^{ALE}\ V_{,i}) \cdot [\tau]\ \tilde{A}_o^{-1}(\tilde{A}_j^{ALE}\ V_{,i})}{(V_{,k} \cdot \tilde{A}_o V_{,k})}\right] = \text{scalar} \end{aligned}$$

B Appendix - B

B.1 Source Code Distribution and Executable Build Environment

Source code with sample input files and user manual has been delivered to WPAFB and WPAFB reserves the rights to distribute the software developed under this contract.

B.2 Building STORESIM

1. Change directory to `scripts` '`$ cd scripts`'
   - Look at the file `sysdep`. This file contains all system dependent information. Carefully go over the various compilers, options and paths specified here and make corresponding modifications for your system. At the minimum, you will need to modify the variable ROOT which should be set to the pathname for storesim directory (i.e., the path that you get by doing 'pwd' in storesim directory).
   - Check the path specified for PERL. If it needs to be modified, then make the change in sysdep and also make similar change in file w4 (the first line).
2. All the makefiles in storesim are based on GNU's make. Therefore, a copy of this make is also included in the scripts directory by the name of `makestore_xxx`. You must include the path to this executable in your shell variable `path` (in your .cshrc/.tcshrc file).
3. To make the code, go to the storesim directory and just say `makestore_xxx`.

B.3 Running STORESIM

Please refer to the accompanying User Manual.

C Appendix - C

C.1 User Manual

STORESIM/TETMESH Training Session

User Manual (Quick Reference)

STORESIM/TETMESH User Manual

1.0 Introduction:

This manual explains the basic steps necessary to generate tetrahedral meshes using TETMESH and perform the flow analysis using STORESIM. TETMESH can also be run from the main menu of STORESIM. Please note that this is a preliminary version of the manual and does not explain all the features. A complete description of all features and procedures to select various options will be made available in future releases.

In the following. several commands have been described that invoke various modules. In the definition of those commands the variable $STORESIM_DIR needs to be set using the standard setenv command and its value is the path name to the storesim (root) directory. For example to run STORESIM, the instruction would be to execute $STORESIM_DIR/storesim.

2.0 Input Geometry Definition:

TETMESH accepts geometry information in the form of both IGES trimmed surface format and TETMESH E3S trimmed surface format. In addition, TETMESH is capable of converting Network (or panel code) format to TETMESH E3S trimmed surface format.

2.1 Procedure to Convert a Network File to E3S File:

In this section we explain the steps involved in converting a Network (or panel code) format file to TETMESH E3S trimmed surface format. First, TETMESH processes data from a Network file (say *model.net)* and creates a file *net2e2s.* The data in this file is in E2S format and needs to be converted to E3S format. *e2s−to−e3s.pl* processes the contents in *net2e2s* file and outputs the data in E3S format. The necessary commands to convert the data from Network format to E3S format are given below.

a. To convert the model.net to net2e2s format type: **$STORESIM_DIR/tet −N model.net**

b. To convert net2e2s to model.e3s type: **$STORESIM_DIR/e2s−to−e3s.pl < net2e2s > model.e3s**

2.2 Sample Network file:

This file contains the surface information in the form of a set of (m x n) xyz coordinates for each surface. This does not allow for trimmed surfaces. Description of a sample 'Net' file is given below:

<Title>
Number of Surfaces
<Number of surfaces>
$<n_{i-1}> <n_{j-1}>$ <surface>
x(1,1) y(1,1) z(1,1)
:   :   :
$x(n_{i-1},1)\ y(n_{i-1},1)\ z(n_{i-1},1)$
:   :   :
$x(n_{i-1},n_{j-1})\ y(n_{i-1},n_{j-1})\ z(n_{i-1},n_{j-1})$

Next Surface
: : :

Sample Network file:

```
Sphere within a sphere
Number of Surfaces
16
10 10 inner-0
1.000000 0.000000 0.000000
0.984808 0.173648 0.000000
   :                     :         :
0.766044 0.642788 0.000000
0.642788 0.766044 0.000000
   :                     :         :
10 10 inner-1
   :                     :         :
10 10 inner-16
0.000000 1.000000 0.000000
-0.173648 0.984808 0.000000
-0.342020 0.939693 0.000000
   :                     :         :
```

2.3 Sample E3S file:

The E3S format is an internal format. As explained in the previous section , the Network files can be converted into E3S format to perform the mesh generation and flow analysis . In E3S format, the geometry is defined by **support–surfaces**, **edges**, and **trimmed–surfaces**. The support–surfaces are defined by wireframes of (m x n) points. A set of xyz–points defining a trimming curve denotes an edge. A trimmed–surface is defined by a set of boundaries (a boundary is a set of edges). The flow boundary conditions, component definition, material properties and component trajectory information is appended to complete the E3S format file. A sample E3S file is given below.

```
/* support surfaces */
{SUPPORT-SURFACE ss1
  {BILINEAR 10 10
    {
      1 0 0
      0.984808 0.173648 0
        :    :    :
      0 0 1
    }
  }
}
/* boundary segments */
{TRIMMED-SURFACE inner-0 ss1 +OUTWARD-NORMAL
  {
    { +e1 -e2 +e3 }
  }
}

/* intersection segments */

{EDGE e1 CURVED inner-0 inner-1
  {NODES 5 {DISTRIBUTION INPUT}}
  {
    0 1 0
    :  :   :
    0 0 1
  }
}
/* walls */
{BOUNDARY-CONDITION bc1 SOLID-WALL-EULER
  {0.00027648 1 0 0 1}  /* (Temperature, u1, u2, u3, Density) */
  {inner-0 inner-1 inner-2 inner-3 inner-4 inner-5 inner-6 inner-7}
}
{COMPONENT s6dof SIXDOF-SOLVER-CALCULATE
  {inner-0 inner-1 inner-2 inner-3 inner-4 inner-5 inner-6 inner-7}
}

#
# The following PROGRAM p1 defines the initial conditions and mass/moment-of-inertia
# properties for the COMPONENT s6dof (this can also be used as a template).
#

PROGRAM p1  [
s6dof_accg  = 9.81;
s6dof_amass = 1.0;
s6dof_aixx = 1.0;
s6dof_aiyy = 1.0;
s6dof_aizz = 1.0;
s6dof_aixy = 0.0;
s6dof_aixz = 0.0;
s6dof_aiyz = 0.0;

 proc s6dof_motion () {
   time       = $1;
   psi_dot     = 0.0;
   theta_dot   = 0.0;
   phi_dot     = 0.0;
   s6dof_xc      = 0;
       s6dof_yc         = 0;
       s6dof_zc         = 0;
       s6dof_vxc        = 5.0;
       s6dof_vyc        = 0.0;
       s6dof_vzc        = 0.0 ;
#
#      YAW and YAW-RATE
#
       s6dof_psi        = 0.0;
       s6dof_psi_dot    = 0.0;
#
#      PITCH and PITCH-RATE
#
       s6dof_theta      = 0.0;
       s6dof_theta_dot = 0.0;
#
#      ROLL and ROLL-RATE
#
       s6dof_phi        = 0.0;
       s6dof_phi_dot    = 0.0;
    };
```

3.0 Graphics Server Setup:

The graphics part of STORESIM/TETMESH is based on the Motif and X11 standards. A proper setup of your workstation requires:

1. Setting the environment variables DISPLAY to the name of your terminal by setenv DISPLAY your_term:0

2. Providing a set of default bindings in the .Xdefaults file. Apart from selecting the fonts, colors and locations of various widgets, the .Xdefaults file also enables the user to move the objects displayed in the viewport. Suggested defaults for the TETMESH are given in Table 1. In most cases, it is suitable to copy the provided .Xdefaults file to your home directory under another file name, and then append this file to your existing .Xdefaults file to preserve the contents of the existing .Xdefaults file. These additional .Xdefaults specification are also repeated in a companion files $STORESIM_DIR/Xdefaults–tet.

3.1 Mouse Button Functions

The user may position the objects displayed in the viewport window by pressing one of the three mouse buttons. The definitions of these buttons corresponding to the .Xdefaults setting shown in Table 1 are:

1. Button 1 (Left) Pan: Move (drag) objects in the window.
2. Button 2 (Middle) Trackball: This mode allows the objects to be rotated in the window.
3. Button 3 (Right) Zoom: Moving the mouse horizontally performs the zoom action, basically enlarging/reducing the size of the picture within the window.

```
! These defaults are for stand alone tet package
!   form names are:
!     output_form, exit_form, edge_editor
!
TETMESH*background:                sgiteal
TETMESH*selectColor:               red
TETMESH*shadowThickness:      2
TETMESH*highlightThickness:   0
TETMESH*imagetet.translations:#override \n\
  <Btn1Down>:         Pan(d)        \n\
  <Btn1Up>:           Pan(u)        \n\
  <Btn1Motion>:       Pan(m)        \n\
  <Btn2Down>:         Trackball(d)  \n\
  <Btn2Up>:           Trackball(u)  \n\
  <Btn2Motion>:       Trackball(m)  \n\
  <Btn3Down>:         Zoom(d)       \n\
  <Btn3Up>:           Zoom(u)       \n\
  <Btn3Motion>:       Zoom(m)       \n\

! Tetmesh control form and viewport
TETMESH*tetview.defaultPosition:   false
TETMESH*tetview.x:  650
TETMESH*tetview.y:  550
TETMESH*tetview.width:  500
TETMESH*tetview.height:  500
TETMESH*tet_form.defaultPosition:   false
TETMESH*tet_form.x:  0
TETMESH*tet_form.y:  0
TETMESH*tet_form.width:  450
```

Table 1: Contents of .Xdefaults file for.

4.0 STORESIM Solver Options:

STORESIM is a multibody CFD flow solver capable of handling moving boundaries and rigid body motions. The **Move Components Form** provides a list of valid logical combinations of solvers that can be used in the simulation.

5.0 Performing a Simulation:

In this section we give a quick step-by-step procedure to run STORESIM/TETMESH. As mentioned in the Introduction, we do not try to explain all existing features and capabilities of the STORESIM/TETMESH program. However, for this beta release, the steps described will enable the user to generate the mesh from an E3S file, run the flow solver and postprocess the solution.

To start the execution type: storesim and hit <return/enter>.

This brings up the main menu with the following options:

Read Files: To read in restart or save files.
Viewport : To postprocess the solution.
Run Storeflow: Execute the STORESIM flow solver.
Run Tetmesh: Execute TETMESH to generate mesh and execute STORESIM flow solver.
Save: To save the solution to a restart file.
Done: End the solver session.

5.1 Read Files:

The Read Files Control form consists of the Standard Motif Selection form. This is used to select and read the solution file for postprocessing. The Read Files Control form is provided with options to read the files in Tetmesh grid file format, Storeflow Solution File format, COBALT .pix file format, and COMCO restart file format.The form consists of a scrollable list of file names, directories, input text field and Select and Filter buttons.

5.2 Viewport:

After loading the solution file, by clicking on the Viewport button a postprocessing window appears on the screen. This window has several options for postprocessing the results. The Control button on the top of this window brings up the post-processor's primary control panel. This provides common options, such as selecting the background color, mesh display, color bar display, 3D projection type, etc. Many of these options are activated by toggle switches. By clicking on the Load Results button different components can be activated for postprocessing. To activate (select) a component, click on the name the of the component. The buttons act as toggle switches and can be also used to deactivate a component. Only active components can be displayed in the viewport.

To activate the slicing plane feature, hit the Slice button on the Viewport menu or the one in the Control Panel. Using this option, scalar data may be visualized by intersecting the grid with planes which are colored according to interpolated values of the desired solution component.

Whereas slicing planes display scalar data according to the geometric location, isosurfaces are defined by the data itself. The loci of points with the same data values are an arbitrarily shaped surface in space, perhaps with disconnected regions.

Velocity vectors button enables the user to plot the velocity vectors at the nodes. These vectors are scaled relative to the maximum normalized velocity in the flow domain to provide an indication of the change in magnitude of the velocity vector at various points in the domain. The remaining buttons displayed at the top of the viewport window are not yet completely functioning for the beta release.

5.3 Run Storeflow:

After loading the geometry file and the mesh, by clicking on this button the user executes the STORESIM solver. The link to this button is not yet properly established for this release. However, for this release, the flow solver can be accessed through the TETMESH. This is explained in the next section.

5.4 Run Tetmesh:

Choosing this option from the STORESIM main menu accesses the TETMESH gird generator. This brings up a standard file selection form to read the geometry information. The file must be in the E3S format and have **.e3s** extension. Once the file has been selected, TETMESH preprocesses the data to verify the format. After successfully reading the data, TETMESH opens the following: a Viewport window to display the object, a Component Parameters widget, and a TETMESH menu widget. In this section we explain the options and describe the steps involved in generating the grid and running the flow solver.

5.4.1 Viewport:

The TETMESH viewport is different from the STORESIM viewport described in Section 5.2 The purpose of the TETEMSH viewport is to display the object (components) and the mesh. On the other hand, the STORESIM viewport is designed to meet additional needs of postprocessing the solution.

5.4.2 TETMESH Menu:

This is the main control form from which the display options, mesh generation option and flow solvers are controlled. At any given time, all active options are highlighted and the non–active options are dimmed (or grayed out). In the control form all options related to a particular task are grouped together and are separated by a dividing line. In the remainder of this section we explain the options.

5.4.2.1 Viewport Options:

Surface Edges: This is a toggle used to display/clear the selected surfaces.

Surface Names: This is a toggle used to print/erase the names of selected surfaces.

Edge Names: This is a toggle used to print/erase the names of selected edges.

Plot Wireframe: This is a toggle used to display/erase the wireframe mesh for the surfaces.

Two–D Mode: Display/Erase the wireframes of selected surfaces in Non–uniform Parameter space (Parametric map of xyz in UV space).

Edge Discretization: Displays the location of the nodes on the selected surfaces.

Triangles: Displays the surface triangles on the selected surfaces.

Nodes: Displays nodes on the selected surfaces.

Non–Unif. UV–Space: (Not Active)

Unenforced Faces: Display all enforced faces.

Normal: Display the surface normals on selected surfaces.

Flip: Reverse the orientations of surface normals of selected surfaces.

Boundary Condition: Print the specified boundary conditions on selected surfaces.

Special Plot: For future use.

Tetrahedra (dynamic): Display all elements with aspect ratio less than the minimum aspect ratio. (If the aspect ratio of all elements is greater than the minimum value, plot the entire mesh.)

Tetrahedra (fast): Display the tetrahedral mesh.

Aspect: Allowable minimum aspect ratio.

Reset: To reset the object orientation in the viewport.

5.4.2.2 Surface Selector:

Surface Selector: This contains a scrollable list of all selectable surfaces. All surfaces can be selected by clicking the left button of the mouse on **All** and by clicking on **De–sel**, all surfaces will be de–selected. All selected components (surfaces) will be highlighted and individual components can be selected/de–selected by clicking the left button of the mouse on the name of the surface.

Surface Spacing: Provides additional options to discretize the surface. (will be explained at the short–course).

5.4.2.3 Edge Selector:

Edges to Employ: This contains a scrollable list of all selectable edges. All edges can be selected by clicking the left button of the mouse on All. By clicking on De–sel all surfaces will be de–selected. All selected edges will be high–lighted and individual components can be selected/de–selected by clicking the left button of the mouse on the name of the edge.

Edge Editor: Provides additional options to discretize the edges. (will be explained at the short–course).

5.4.2.4 Triangulate Surfaces:

Do Everything (Surf): This is a one step option to triangulate all surfaces in the geometry. This first discretizing the edges and then merges the nodes on matches edges. Next all surfaces are triangulated. These steps can be carried out sequentially by selecting the *Discretize Edges* and the options provided under *Triangulate Surfaces*. The usages of these options will be explained at the short–course.

5.4.2.5 Tetrahedralize Regions:Do Everything (Surf): This is a one step option to tetrahedralize all surfaces in the geometry. This first discretizing the edges and then merges the nodes on matches edges. Next all surfaces are triangulated. These steps can be carried out sequentially by selecting the *Discretize Edges* and the options provided under *Triangulate Surfaces*.

5.4.2.6 Solver:

Move Components...: This enables the user to select the type of solver, time integration scheme, and other solver parameters. The **Move Components** button in this widget fires the appropriate solver.

Smooth Mesh...: This widget provides the user with different options for selecting the mesh smoothing parameters. In this release only Node Flow algorithm is used for computing the node velocities and updating the mesh.

Flow Params...: The reference (farfield/inflow) flow parameters are displayed in this widget. Also some run control parameters (CFL and number of time steps) are specified here.

Init Flowfield: This button is used to initialize the flow field to inflow conditions. The existing solution (if any) will be lost and the flow is initialized to inflow values on the entire flow domain.

Fire Storeflow: Only the flow solver (without mesh movement and rigid body motion) is executed if the user clicks on this button.

5.4.2.7 Output Control:

This form can be used to save the database into E3S format or COMCO restart format. The COMCO restart file format can then be used to postprocess the solution.

5.5 Save:

Choosing the Save option from the STORESIM main menu accesses the Save form, which is used for writing a copy of the data base to a user–prescribed file. This file can be used to post–process the solution and/or to continue the execution of the flow simulation.

5.6 Done:

Chosing the Done option from the STORESIM main menu the session can be stopped.

Run Tetmesh: Once the data is stored in the entered/converted to E3S format

6.0 TETMESH Command Line Options:

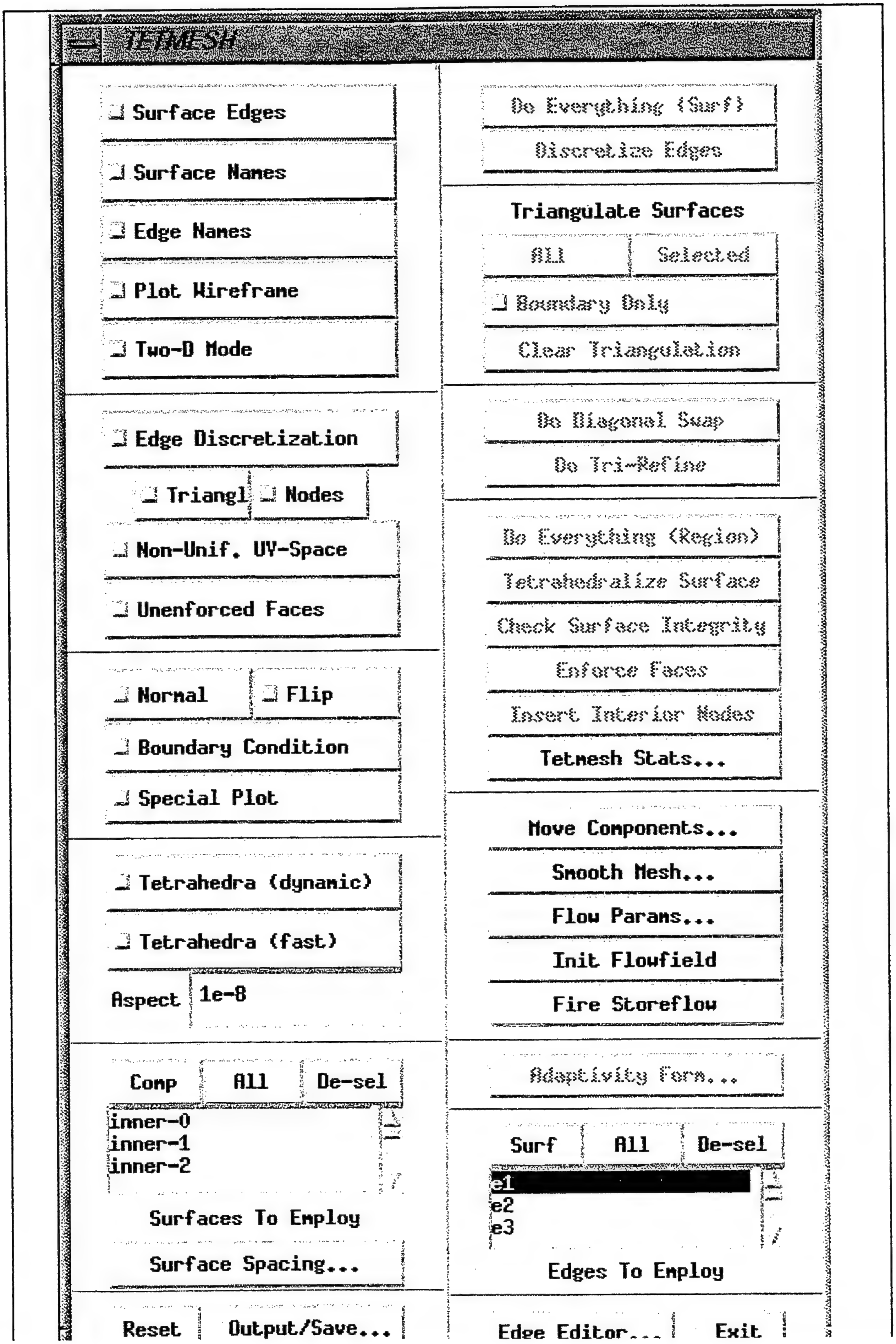
The valid options to run stand alone version of TETMESH are. These options are self explanatory. The syntax to execute stand alone TETMESH is : **STORESIM_DIR/tet [option] file.e3s [option]**

```
Valid options ([short-form] [long-form]):
     [-A] [--allow-epsilon-boundary-node-movement]
     [-a x] [--aspect-ratio-threshold=x]
     [-b] [--batch]
     [-B] [--brute-force-edge-check]
     [-c x] [--curvature-measure=x]
     [-d] [--dump-boundary-points]
     [-e x] [--edge-delta=x]
     [-f x] [--flatness-measure=x]
     [-g] [--gintern3-debug]
     [-h] [--help]
     [-i x] [--max-interior-nodes=x]
     [-I] [--interactive]
     [-n x] [--max-nodes=x]
     [-o x] [--output-basename=x]
     [-O x] [--solver-option=x]
     [-r x] [--radius-factor=x]
     [-F x] [--output-format=x]
     [-s x] [--tet-shrink-factor=x]
     [-Q x] [--auto-save=x]
     [-q x] [--auto-save-file=x]
     [-R x] [--restart-file=x]
     [-S x] [--restart-step=x]
     [-C x] [--restart-codeID=x]
     [-t x] [--tolerance-point-same=x]
     [-T] [--fp3-rejection-test-off]
     [-V] [--verbose]
     [-v] [--version]
     [-N] [--net-to-e2s]
```

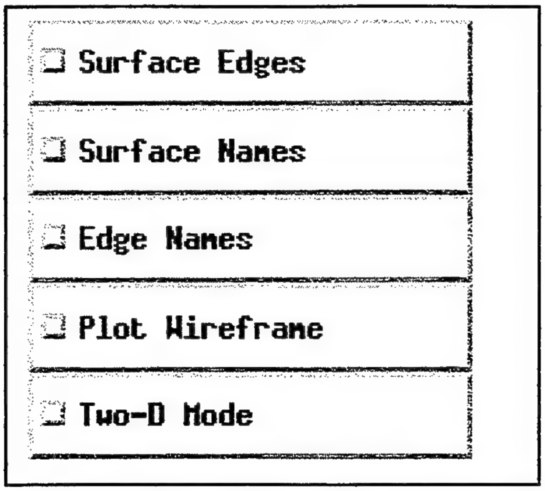
7. Input Parameters:

Fluid properties and run control parameters can be input to the program through an input file. The default fluid properties and other job control parameters (solver option, time step scheme, restructure frequency etc.,) are input through the file **'.StoresimDefaults'.** These values can also be modified in the widgets. All derived quantities such as the Mach No., Reynolds, No. cannot be modified, instead they are recomputed internally and displayed in the widget. To make the program read the ".StoresimDefaults" file, before executing the job you need to type: setenv `COMCODEFAULTSFILE .StoresimDefaults`
A sample .StoresimDefaults is given below.

```
JobName                       spheres.e3s
Title                         Flow over two spheres
Fluid_Name                    AIR
Ref_Density                   1.225
Ref_Viscosity                 1.7876E-05
Ref_Temp                      288.15
Ref_Velocity                  1020.8823
Ref_Length                    1.0
Gamma                         1.4
GasConstant                   287.05307
SpecificHeat                  1004.675
Conductivity                  0.02624
Sutherland_Const_T1           110.0
Sutherland_Const_T2           120.0
cfl                           0.3
nsteps_initial                100
nsteps_simulation             10
Solver_Option                 2
Time_Integ_Scheme             1
Use_Local_Or_Global_Time_Step 0
Restructure_Frequency         5
Total_Time_Steps              50
Adaptation_Frequency          0
Save_Step_Frequency           25
Max_Mem_In_A_Step             0
Delta_T                       0.025
Invoke_Solver_Frequency       1
Pseudo_CFL                    0.2
Pseudo_Mach                   0.4
Pseudo_Reynolds               0.5
```


TETMESH
☐ Surface Edges
☐ Surface Names
☐ Edge Names
☐ Plot Wireframe
☐ Two-D Mode
☐ Edge Discretization
☐ Triangl
☐ Nodes
☐ Non-Unif. UV-Space
☐ Unenforced Faces
☐ Normal
☐ Flip
☐ Boundary Condition
☐ Special Plot
☐ Tetrahedra (dynamic)
☐ Tetrahedra (fast)
Comp
All
De-sel
inner-0
inner-1
inner-2
Surfaces To Employ
Surface Spacing...
Reset
Output/Save...
Do Everything (Surf)
Discretize Edges
Triangulate Surfaces
All
Selected
☐ Boundary Only
Clear Triangulation
Do Diagonal Swap
Do Tri-Refine
Do Everything (Region)
Tetrahedralize Surface
Check Surface Integrity
Enforce Faces
Insert Interior Nodes
Tetmesh Stats...
Move Components...
Smooth Mesh...
Flow Params...
Init Flowfield
Fire Storeflow
Adaptivity Form...
Surf
All
De-sel
e1
e2
e3
Edges To Employ
Edge Editor...
Exit

Aspect 1e-8



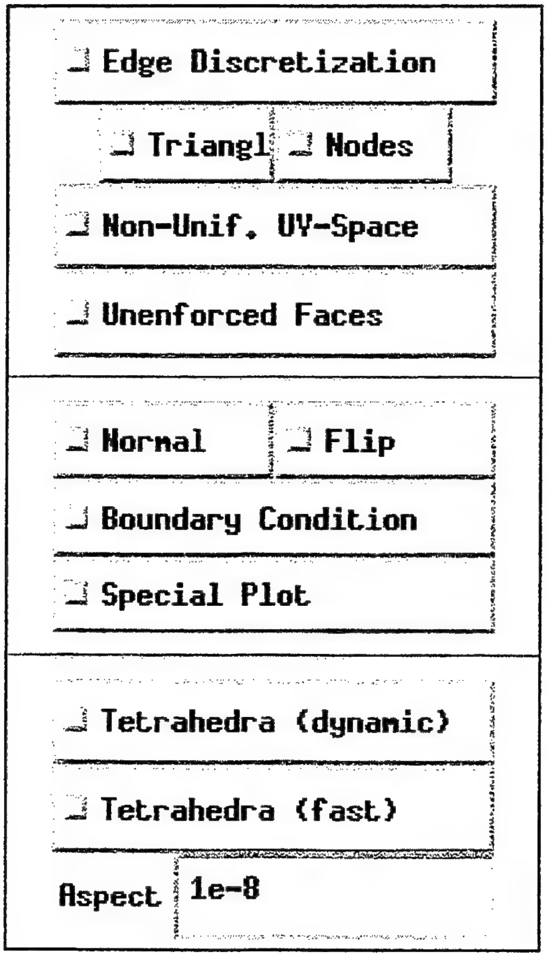
Display the edges of selected surfaces

Display the names of selected surfaces.

Display the edge names of selected surfaces.

Display the wireframe mesh for selected durfaces.

Plot the wireframes of selected surfaces in non-uniform parameter space.



Display the edge discretization.

Display surface triangles/nodes.

Not Used.

Display unenforced faces.

Display surface normals.

Reverse the orientations of the surface normals of selected surfaces.

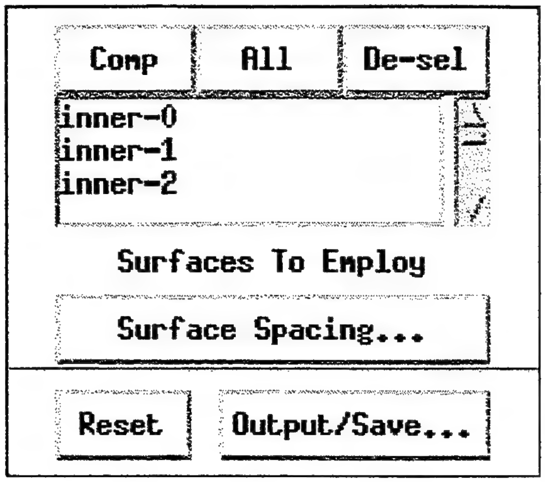
Display the boundary conditions.

Not Used

Display all elements with aspect ratio < the value given for **Aspect** (If all elements have aspect ratio > **Aspect**, all elements get displayed).

Display all elements.

Minimum allowable aspect ratio.



Select all surfaces in a given component.

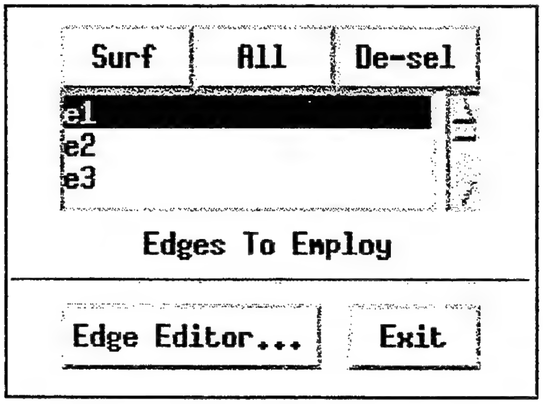
Select all surfaces in the geometry.

De-select all selections.

Open the surface spacing parameter editor.

Reset the graphics viewport to default position.

Open the Output/Save control form.



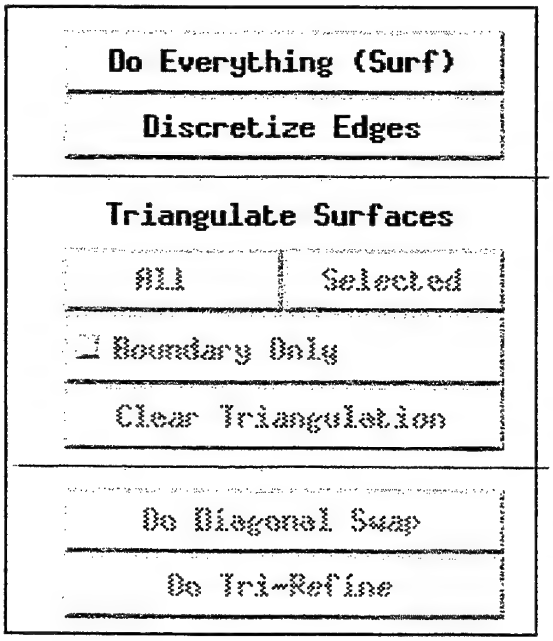
Select all edges in a selected component.

Select all edges in the geometry.

De-select all selections.

Open the edge editor widget.

Exit from the TETMESH.



One step option to triangulate the surfaces.

Discretize the edges. (To define the node distribution on the edges use the Edge Editor before discretizing the edges).

Triangulate all surfaces.

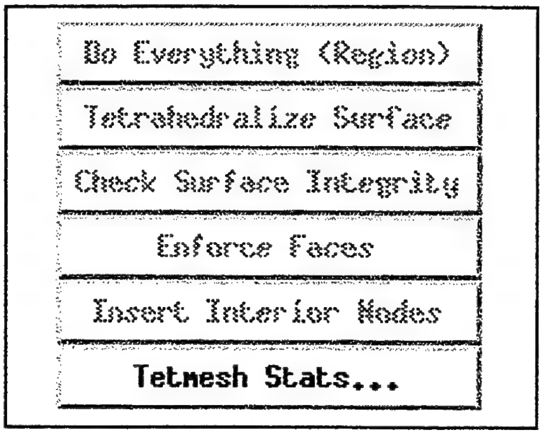
Triangulate only selected surfaces.

Triangulate only boundaries (edges).

Clear the triangulation.

Do diagonal swapping to produce Delaunay mesh.

Does some data structure setup for tetrahedralization.



One step option totetrahedralize all regions.

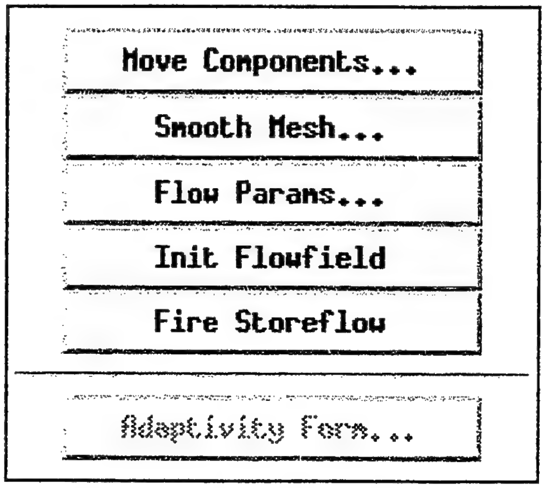
Tetrahedralize the surfaces.

Check if the faces of surfaces match properly with neighbors.

Not Used.

Inset interior nodes and create tetrahedral elements.

Open the Statistics widget.



Open the menu to run the six-dof solver.

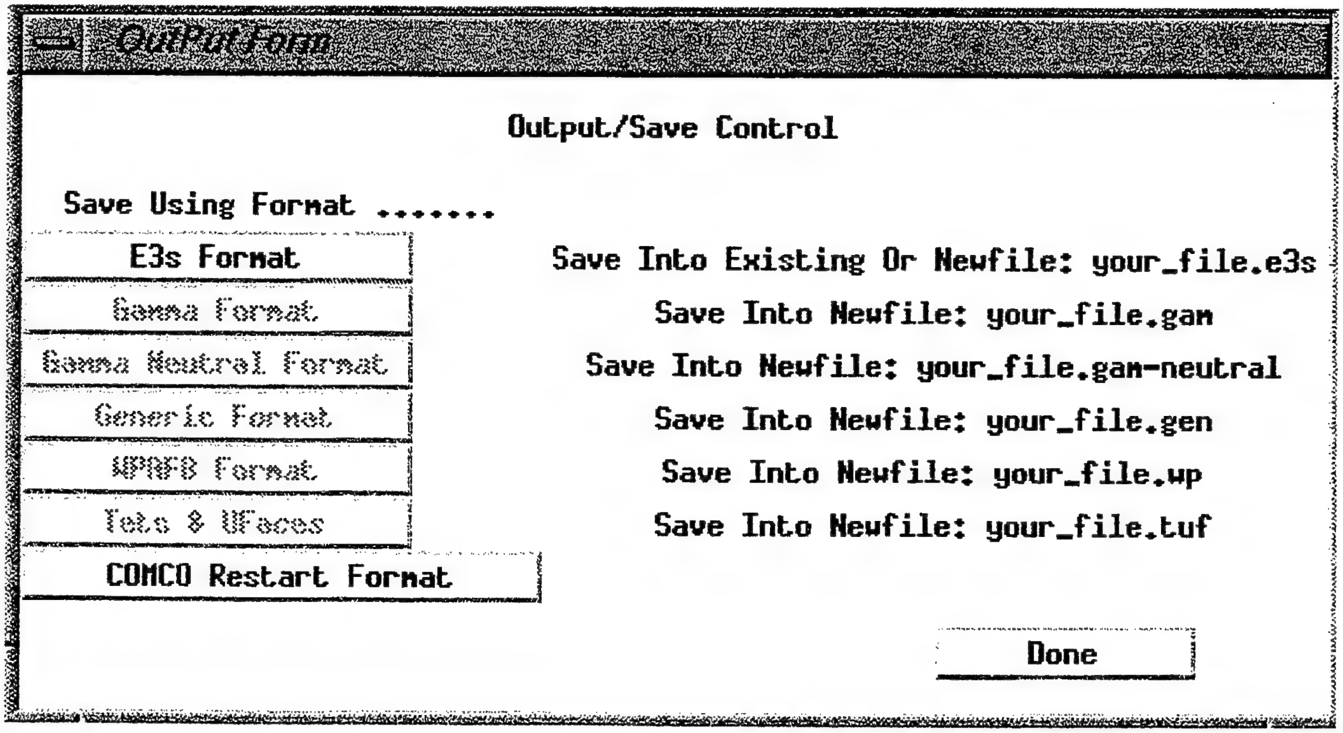
Open the menu to set parameters for mesh smoother.

Display the flow parameters.

Initialize the flow to inflow conditions.

Execute the flow solver.

Not Used.



Save Into Existing Or Newfile: your_file.e3s

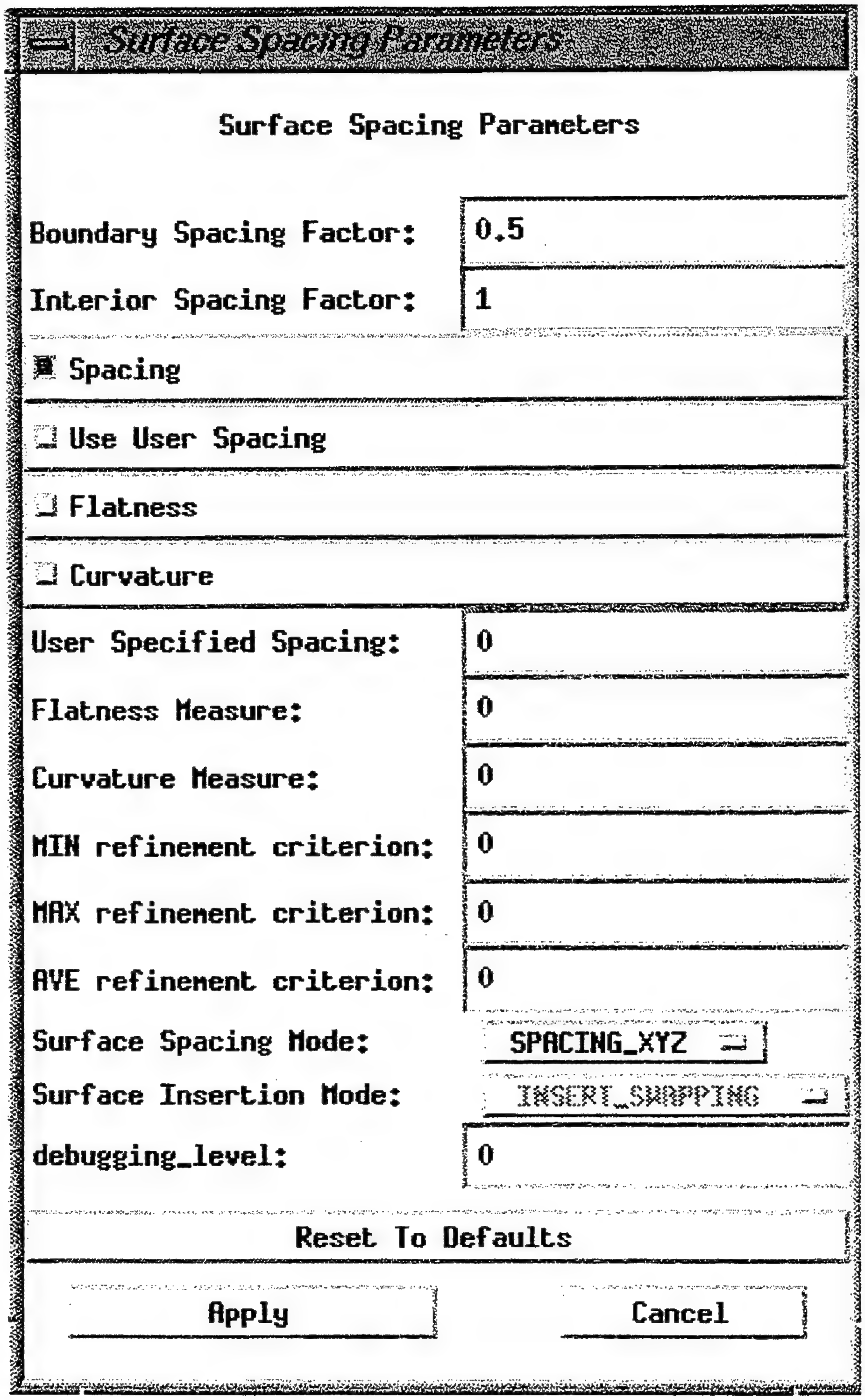
Save Into Newfile: your_file.gam

Save Into Newfile: your_file.gam-neutral

Save Into Newfile: your_file.gen

Save Into Newfile: your_file.wp

Save Into Newfile: your_file.tuf



Surface Spacing Parameters

Boundary Spacing Factor: 0.5

Interior Spacing Factor: 1

■ Spacing

□ Use User Spacing

□ Flatness

□ Curvature

User Specified Spacing: 0

Flatness Measure: 0

Curvature Measure: 0

MIN refinement criterion: 0

MAX refinement criterion: 0

AVE refinement criterion: 0

Surface Spacing Mode: SPACING_XYZ

Surface Insertion Mode: INSERT_SWAPPING

debugging_level: 0

Parameters to control the triangle density at the edges and on the surfaces. (Smaller value produces more triangles)

When selected uses above parameters.

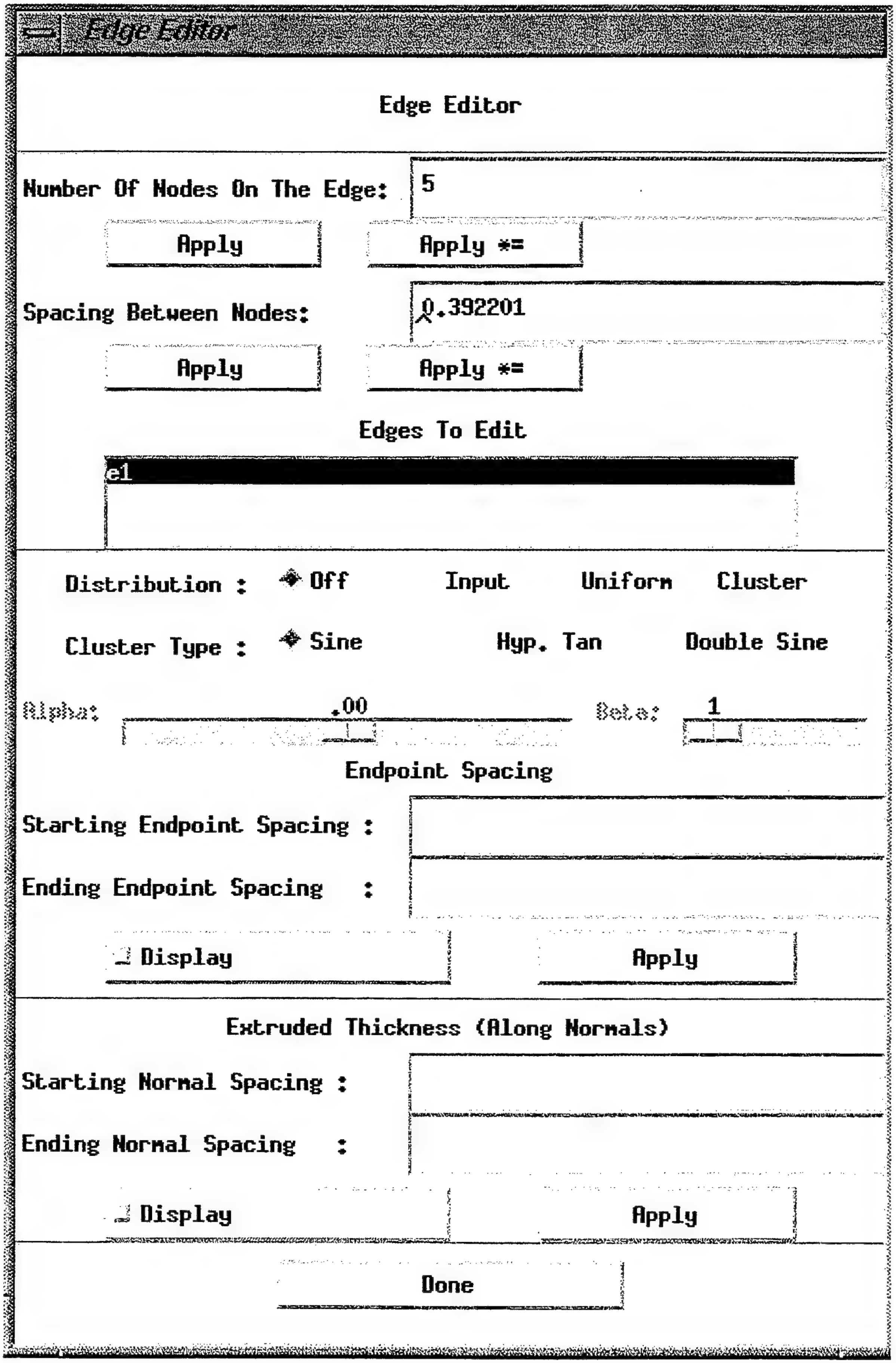
The triangulation can also be done using any combinations of these methods. When using these, input a non-zero number (appropriately chosen) for the corresponding measure.

These are READ ONLY

Valid Options are UV, XYZ, and XYZ_2

Do not accept any changes

Accept the entered changes
Close the widget.



Edge Editor

Number Of Nodes On The Edge: 5

Spacing Between Nodes: 0.392201

Edges To Edit

e1

Distribution : ◆ Off Input Uniform Cluster

Cluster Type : ◆ Sine Hyp. Tan Double Sine

Alpha: .00 Beta: 1

Endpoint Spacing

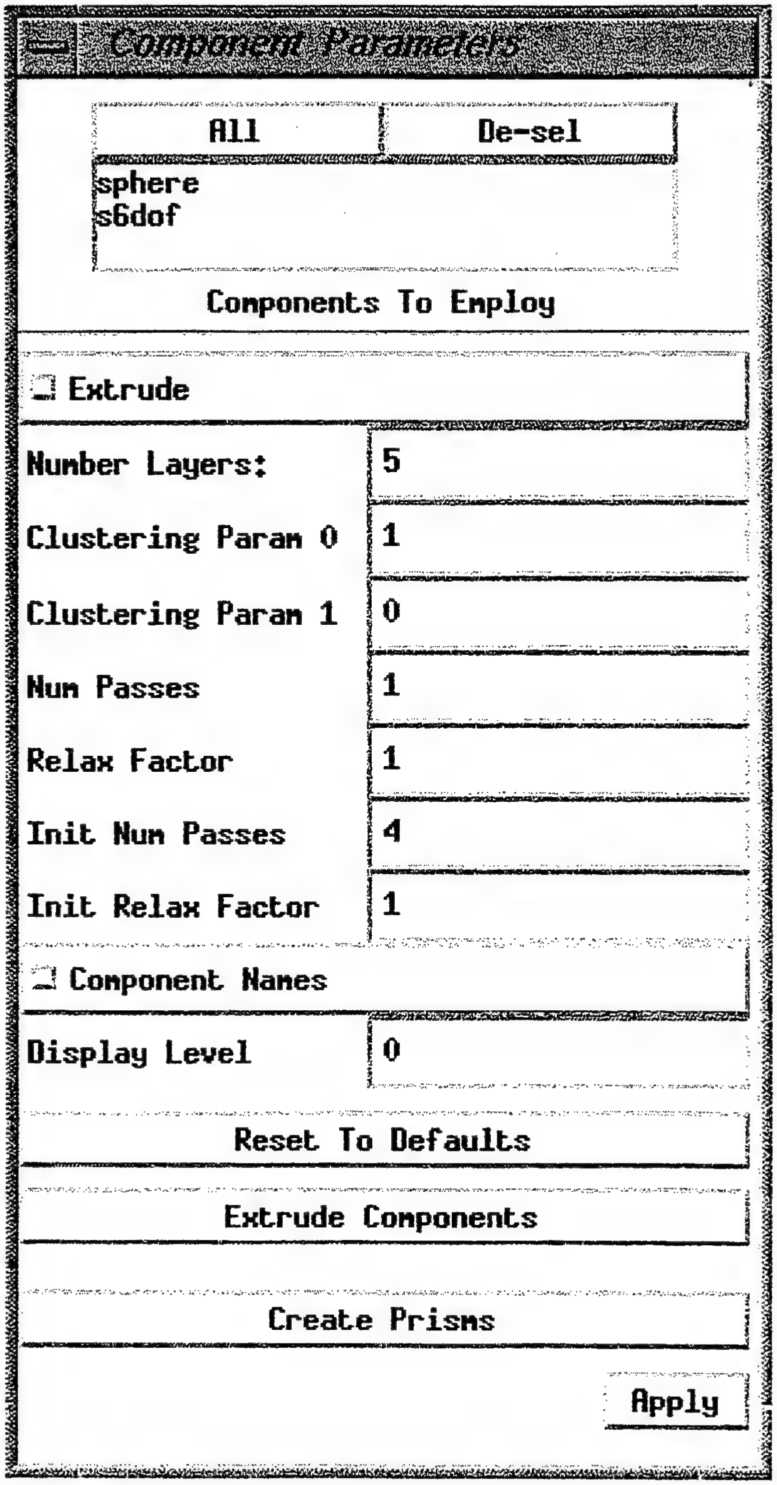
Starting Endpoint Spacing :

Ending Endpoint Spacing :

Extruded Thickness (Along Normals)

Starting Normal Spacing :

Ending Normal Spacing :



Number Layers: 5

Clustering Param 0: 1

Clustering Param 1: 0

Num Passes: 1

Relax Factor: 1

Init Num Passes: 4

Init Relax Factor: 1

Display Level: 0

Component selector.
All: To select all components
De-sel: To de-select all components.
To select individual component, click on its name.

Extrude ON/OFF button

Number of layers of elements to be put in extruded region.

Weighting factor for biasing the layers.

Number of passes and relax. factor for normal calculations.

Relax. factor and number of passes for averaging the normals.

Not Used

To display a given layer.

Do not accept changes.

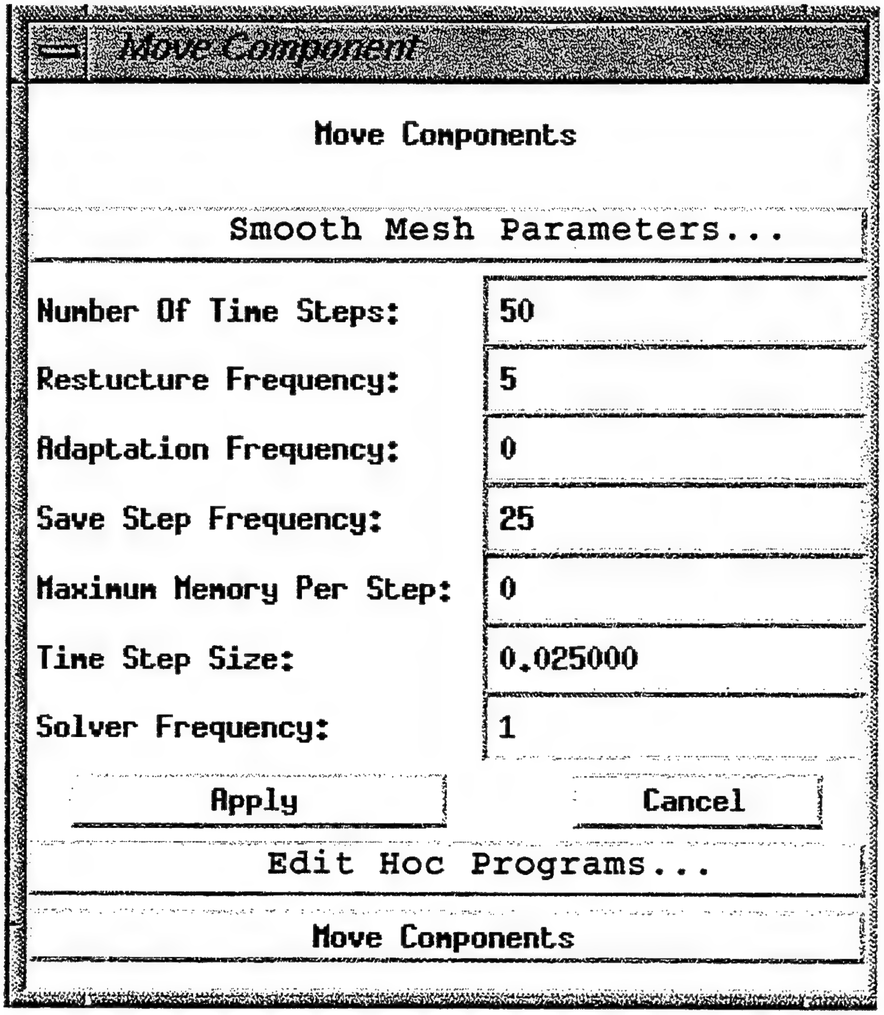
Action button to extrude component(s).

Action button to create prisms.

Accept the entered changes.

To extrude, follow this sequence: Select a component – Exturde ON – Apply – Extrude Components

To display a layer: Enter layer number in DIsplay level and click on Apply.



Number Of Time Steps: 50

Restucture Frequency: 5

Adaptation Frequency: 0

Save Step Frequency: 25

Maximum Memory Per Step: 0

Time Step Size: 0.025000

Solver Frequency: 1

Display the mesh smoothing parameters widget.

Max. number of time steps

Frequency of time steps for mesh restructuring.

Not Used

Frequency of time steps for saving the solution.

Not Used

Not Used

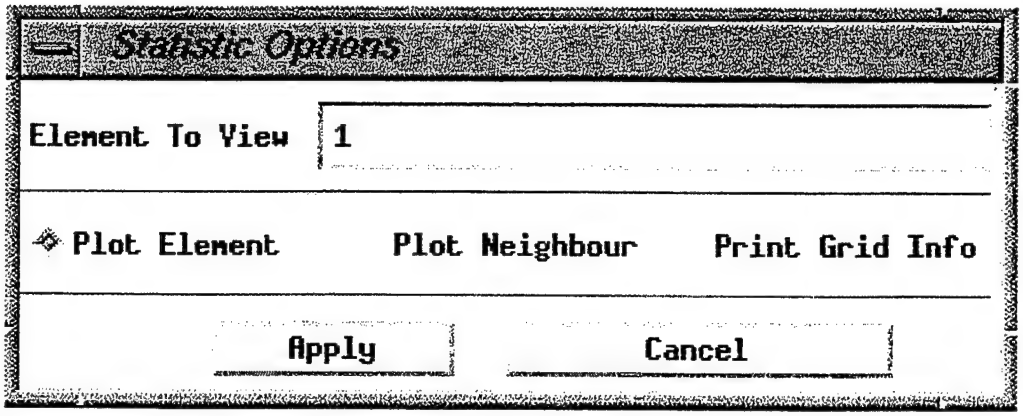
Time stpe frequence for invoking flow solver.

Accept the entered changes.

Close the widget

Open the HOC editor.

Action button to execute the solver.



Element To View 1

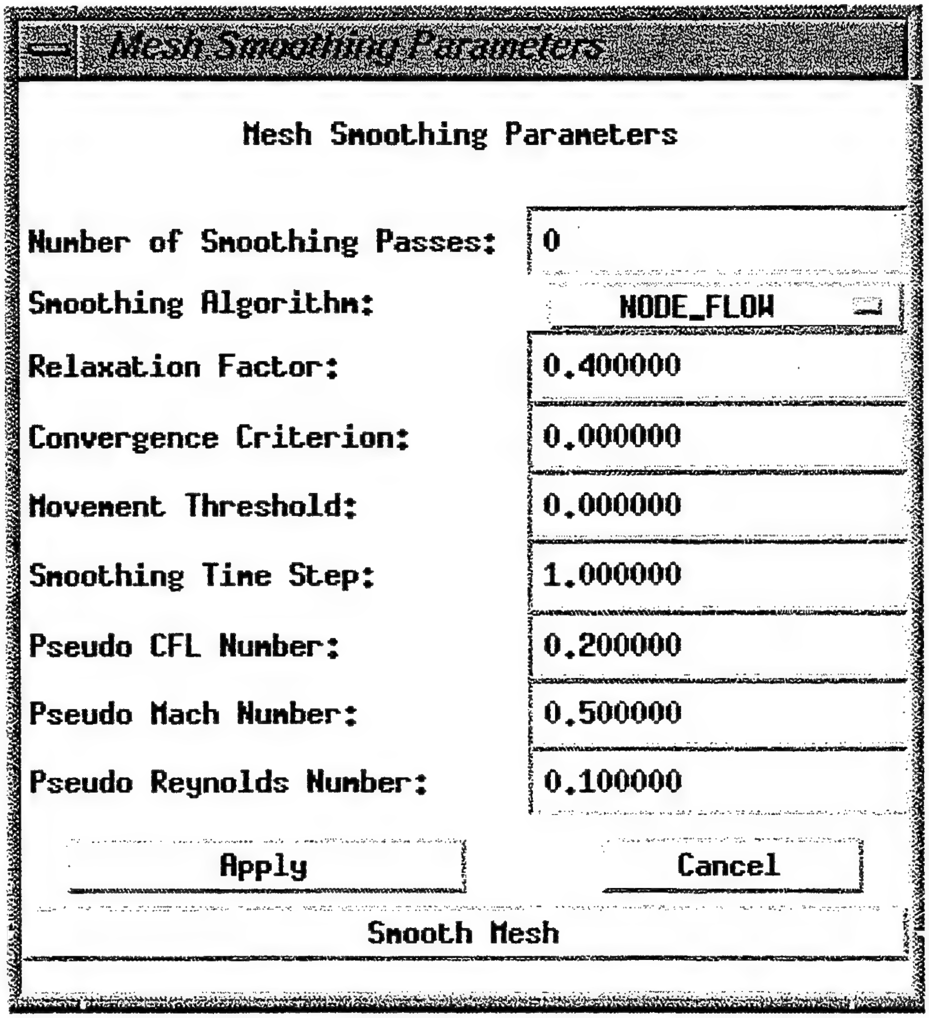
◈ Plot Element Plot Neighbour Print Grid Info

Elem. number

Plot/Print Options

Accept the input.

Cloe the widget



Mesh Smoothing Parameters

Number of Smoothing Passes: 0 *

Smoothing Algorithm: NODE_FLOW — Move the nodes using an incomp. flow analogy

Relaxation Factor: 0.400000 *

Convergence Criterion: 0.000000 *

Movement Threshold: 0.000000 *

Smoothing Time Step: 1.000000 *

Pseudo CFL Number: 0.200000

Pseudo Mach Number: 0.500000

Pseudo Reynolds Number: 0.100000

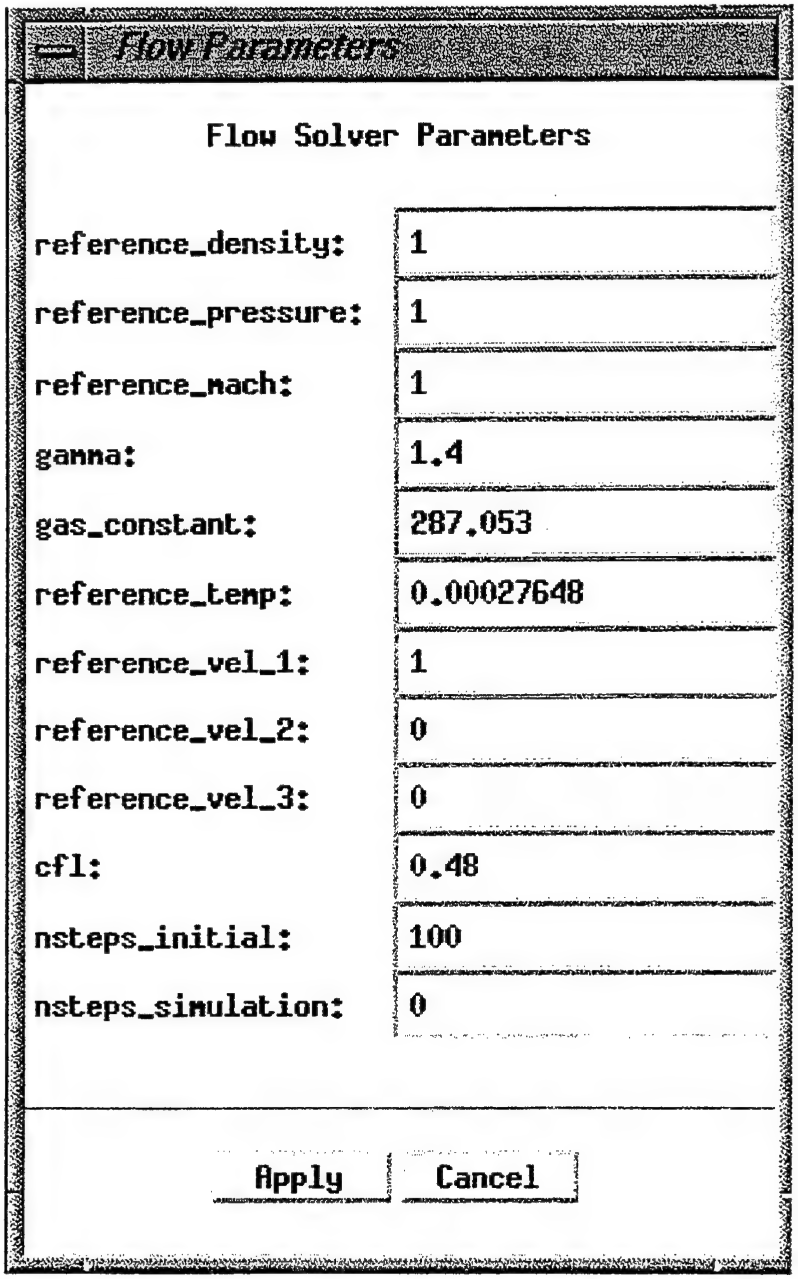
Used for solving the node movement using incompressible flow analogy

Apply — Accept the entered changes

Cancel — Close this widget

Smooth Mesh *

*** Used for stand alone version of mesh smoother only.**



reference_density:	1	Inflow Density (Kg/m^3)
reference_pressure:	1	Pressure at Inflow (N/m^2)
reference_mach:	1	Inflow Mach Number
gamma:	1.4	Ratio of Specific Heats
gas_constant:	287.053	Gas Constant (J/Kg/K)
reference_temp:	0.00027648	Inflow Temperature (K)
reference_vel_1:	1	Inflow X-Velocity (m/s)
reference_vel_2:	0	Inflow Y-Velocity (m/s)
reference_vel_3:	0	Inflow Z-Velocity (m/s)
cfl:	0.48	CFL Number (Use <0.5 for better convergence
nsteps_initial:	100	Number of time steps for flow solver.
nsteps_simulation:	0	Not Used

Apply (Accept) the entered changes

Close this widget

Network File Format

<Title>

<ns> (Character string)

<Number_of_Surfaces>

$N_i \; N_j$ *<Surface_Name_1>*

$x(1,1) \quad y(1,1) \quad z(1,1)$

$\vdots \quad \vdots \quad \vdots$

$x(N_i,1) \quad y(N_i,1) \quad z(N_i,1)$

$\vdots \quad \vdots \quad \vdots$

$x(N_i,N_j) \quad y(N_i,N_j) \quad z(N_i,N_j)$

Repeat Above format for Remaining Surfaces

Support Surface Definition

```
{SUPPORT–SURFACE      <support_surface_name>
      {<surface_type>  <n_x>  <n_y>
          {
               x(1,1)   y(1,1)  z(1,1)
               ...   ...    ..
               x(n_x, 1)   y(n_x,1)  z(n_x,1)
               ...  ...   ...
               x(n_x,n_y)  y(n_x,n_y)  z(n_x,n_z)
          }
      }
}
```

Valid options for surface_type: PLANAR, BILINEAR, IGES–128

Typical Example

```
{SUPPORT–SURFACE  ss1
 {BILINEAR 10 10
  {
    1 0 0
    0.984808 0.173648 0
    0.939693 0.34202 0
    0.866025 0.5 0
```

Support Surface Definition

```
        0.766044 0.642788 0
        0.642788 0.766044 0
        0.5 0.866025 0
        0.34202 0.939693 0
        0.173648 0.984808 0
        0 1 0
         :  : :
        0 0 1
        0 0 1
        0 0 1
        0 0 1
        0 0 1
        0 0 1
        0 0 1
        0 0 1
        0 0 1
        0 0 1
      }
     }
    }
```

Edge Definition (E3S Format)

```
{EDGE <edge–name>   <edge_type> <surface–name–1> [<surface–name–2> ...]]
      {NODES [<number_of_nodes>] <node_distribution_type [<param 1> <param2>] }}
 {
   x_1 y_1 z_1   ! begin list of edge knots
   ...
   x_n y_n z_n   ! n = number of knots
 }
  [iges–description]
  [extrusion–description]
}
Valid Options for edge_type : CURVED, STRAIGHT
Valid Options for node_dist_type: INPUT, UNIFORM, CLUSTER, CLUSTER_HYP_TAN
```

Typical Example

```
{EDGE e1 CURVED inner–0 inner–1
  {NODES 5 {DISTRIBUTION INPUT}}
 {
  0 1 0
     :: :: :: :
  0 0 1
 }
}
```

Edge Definition (contd.)

Example 2: (<number–nodes> = n, so nothing specified for <number–nodes>)

```
{EDGE e3 CURVED ts1 ts6
  {NODES {DISTRIBUTION CLUSTER_HYP_TAN 0.05 0.3}}
 {
   9.4 1.77 6.5
   9.4 1.77 6.47671
   9.4 1.77 6.40251
   9.4 1.77 6.18386
   9.4 1.77 5.68709
  ::  ::  ::  ::
   9.4 1.77 –4.0625
   9.4 1.77 –4.875
   9.4 1.77 –5.6875
   9.4 1.77 –6.5
 }
}
```

Edge Extrusion (E3S Format)

E3S FORMAT: Edge extrusion

```
{EDGE e1 CURVED inner–0 inner–1
  {NODES 5 {DISTRIBUTION INPUT}}
  {
   0 1 0
   0 0.984808 0.173648
   0 0.939693 0.34202
   0 0.866025 0.5
   0 0.766044 0.642788
   0 0.642788 0.766044
   0 0.5 0.866025
   0 0.34202 0.939693
   0 0.173648 0.984808
   0 0 1
  }
  < EXTRUDE LINEAR 0.200000003 0.200000003 >
}
```

Trimmed Surface Definition

```
{TRIMMED–SURFACE <surf_name> <support_surf_name> <Normal_dir> [Surf_type]
    {
        {List of edges with orientation (+/–)}
    }
}
```

Valid Options for Normal_dir: +OUTWARD–NORMAL, –OUTWARD–NORMAL
Valid Options for Surf_type: ZERO–THICKNESS

Typical Examples

```
{TRIMMED–SURFACE inner–0 ss1 +OUTWARD–NORMAL
 {
  { +e1 –e2 +e3 }
 }
}

{TRIMMED–SURFACE patch–4a ss7 +OUTWARD–NORMAL   ZERO–THICKNESS
 {
  { +e6 +e14 –e15 –e16 }
 }
}
```

Trimmed Surface Definition (contd.)

```
{TRIMMED–SURFACE  ts5  ss5  +OUTWARD–NORMAL
 {
   { +e3 +e12 –e7 –e11 }
   {ZERO–AREA –e16 }
 }
}
```

Component Definitions

```
{COMPONENT    <component_name> <Motion_type>
    {List of applicable surfaces given as <surface_name_1>  [<surface_name_2 >...]}
}
```

Valid options for Motion_type are:

NO_MOTION –	No Motion (only flow solver)
SIXDOF–SOLVER–CALCULATE –	Coupled flow, six–dof, and mesh mover
TRAJECTORY–SPECIFIED –	Coupled six–dof and mesh mover

Typical Examples

```
{COMPONENT  sphere  NO–MOTION
 {inner–0 inner–1 inner–2 inner–3 inner–4 inner–5 inner–6 inner–7}
}

{COMPONENT   s6dof   SIXDOF–SOLVER–CALCULATE
 {inner–0 inner–1 inner–2 inner–3 inner–4 inner–5 inner–6 inner–7}
}
```

Boundary Condition Specification

```
{BOUNDARY-CONDITION  <bc_tag_name>  <bc_type>
        {<Temperature/HeatFlux> <X-Velocity> <V-Velocity> <Z-Velocity> <Density>}
        {List of applicable surfaces given as [<surface_name_1>  ....]}
```

```
Valid options for bc_type are:  SYMMETRY, FARSTREAM-CHARACTERISTIC,
                                FARSTREAM-FREE, FARSTREAM-FIXED,
                                SOLID-WALL-ISOTHERMAL, SOLID-WALL-HEAT-FLUX,
                                SOLID-WALL-EULER
```

Typical Examples

```
/* walls */
{BOUNDARY-CONDITION bc1 SOLID-WALL-EULER
 {0.00027648 1 0 0 1}
 {inner-0 inner-1 inner-2 inner-3 inner-4 inner-5 inner-6 inner-7}
}
/* far-field */
{BOUNDARY-CONDITION bc2 FARSTREAM-CHARACTERISITC
 {0.00027648 1 0 0 1}
 {outer-0 outer-1 outer-2 outer-3 outer-4 outer-5 outer-6 outer-7}
}
```

Token map for E3S format:

TOKEN	Matching Name in BNF Grammar
"{"	LBRACE
"}"	RBRACE
"EDGE"	EDGE
"EXTRUDE"	EXTRUDE
"LINEAR"	LINEAR
"SUPPORT–SURFACE"	SUPPORT_SURFACE
"TRIMMED–SURFACE"	TRIMMED_SURFACE
"BOUNDARY–CONDITION"	BOUNDARY_CONDITION
"INFLOW"	YY_BC_INFLOW
"OUTFLOW"	YY_BC_OUTFLOW
"SOLID–WALL"	YY_BC_SOLID_WALL
"WPAFB–BC"	YY_BC_WPAFB
"STRAIGHT"	STRAIGHT_EDGE
"CURVED"	CURVED_EDGE
"IGES–126"	IGES_126_EDGE
"PLANAR"	PLANAR
"BILINEAR"	BILINEAR
"ANALYTIC"	ANALYTIC
"SPLINE"	SPLINE
"BSPLINE"	BSPLINE
"IGES–128"	IGES_128_SURFACE
"NODES"	NODES

"DISTRIBUTION"	DISTRIBUTION
"INPUT"	INPUT
"UNIFORM"	UNIFORM
"CLUSTER"	CLUSTER
"CLUSTER_HYP_TAN"	CLUSTER_HYP_TAN
"CLUSTER_DOUBLE_SIN"	CLUSTER_DOUBLE_SIN
"+OUTWARD–NORMAL"	POSITIVE_OUTWARD_NORMAL
"–OUTWARD–NORMAL"	NEGATIVE_OUTWARD_NORMAL
"COMPONENT"	COMPONENT
"NO–MOTION"	NO_MOTION
"TRAJECTORY–SPECIFIED"	TRAJECTORY_SPECIFIED
"SIXDOF–SOLVER–CALCULATE"	SIXDOF_SOLVER_CALCULATE
"ZERO–THICKNESS"	YY_ZERO_THICKNESS
"ZERO–AREA"	YY_ZERO_AREA
"KEEP–INTERIOR–TETS"	KEEP_INTERIOR_TETS
"IGES–126–DATA"	IGES_126_DATA
"IGES–128–DATA"	IGES_128_DATA
"BASIS–KNOTS"	BASIS_KNOTS
"BASIS–KNOTS–S"	BASIS_KNOTS_S
"BASIS–KNOTS–T"	BASIS_KNOTS_T
"SIZE"	SIZE
"SIZES"	SIZES
"DEGREE"	DEGREE
"DEGREES"	DEGREES
"WEIGHTS"	WEIGHTS
"KNOTS"	KNOTS
"PROP"	PROP
"PARAM"	PARAM

"DEBUG–ON"	
"DEBUG–OFF"	
"–"	MINUS_SIGN
"+"	PLUS_SIGN
"<"	LEFT_ANGLE_BRACKET
">"	RIGHT_ANGLE_BRACKET
Special Cases:	
IDENTIFIER	String of characters, digits, starting with alpha, can contain "_": [a–zA–Z_][a–zA–Z_0–9–]*
COMMENT	C–style comment /* <string> */
LITERAL_INTEGER	integer
LITERAL_FLOATING_POINT	floating point number (with optional exponent)

HOC Program Specifics

```
/* Sixdof-solver or trajectory-specified motion procedure template:
  (Replace <cn> with the component name)  */
PROGRAM  template [
#
# Initial center of mass position
#
<cn>_xc0 = 0.0;
<cn>_yc0 = 0.0;
<cn>_zc0 = 0.0;

#  Mass  properties:
# These are assumed for now to be constant.  If mass properties
#   are changing with time, we need to move these to within
#   the _motion procedure, and update the flow-solver-interface.c routine.

<cn>_accg = 1.0;
<cn>_amass  =  1.0;

<cn>_aixx = 1.0;
<cn>_aiyy = 1.0;
<cn>_aizz = 1.0;
<cn>_aixy = 0.0;
<cn>_aixz = 0.0;
<cn>_aiyz = 0.0;

  proc <cn>_motion () {
   time        = $1;
   vel_x       = 0.0;
   vel_y       = 0.0;
   vel_z       = 0.0;
   psi_dot     = 0.0;
   theta_dot   = 0.0;
```

```
     phi_dot      = 0.0;
     <cn>_xc       = <cn>_xc0;
     <cn>_yc       = <cn>_yc0;
     <cn>_zc       = <cn>_zc0;
     <cn>_vxc      = vel_x;
     <cn>_vyc      = vel_y;
     <cn>_vzc      = vel_z;
#
#     YAW and YAW-RATE
#
     <cn>_psi      = 0.0;
     <cn>_psi_dot  = 0.0;
#
#     PITCH and PITCH-RATE
#
     <cn>_theta     = 0.0;
     <cn>_theta_dot = 0.0;
#
#     ROLL and ROLL-RATE
#
     <cn>_phi      = 0.0;
     <cn>_phi_dot  = 0.0;
#
#     THRUST FORCES AND MOMENTS
#       (these are always in the local coordinate system
#       of the body)
#
     force_x_location = -10.0;
     force_y_location = 0.0;
     force_z_location = 0.0;

     <cn>_thrust_force_x  = 1.0 * exp (-time);
     <cn>_thrust_force_y  = 0.0;
     <cn>_thrust_force_z  = 0.0;
```

```
      <cn>_thrust_moment_x  = force_y_location * <cn>_thrust_force_z –
                           force_z_location * <cn>_thrust_force_y ;

      <cn>_thrust_moment_y  = force_x_location * <cn>_thrust_force_z –
                           force_z_location * <cn>_thrust_force_x ;

      <cn>_thrust_moment_z  = force_x_location * <cn>_thrust_force_y –
                           force_y_location * <cn>_thrust_force_x ;
   };

]
*/

/* Here is the trajectory–specified program: */

PROGRAM p1 [

sphere_xc0 = 0.0;
sphere_yc0 = 0.0;
sphere_zc0 = 0.0;

   proc sphere_motion () {
     time          = $1;
     vel_x         = 0.0;
     vel_y         = 0.0;
     vel_z         = 0.0;
     psi_dot       = 0.0;
     theta_dot     = 0.2;
     phi_dot       = 0.0;
     sphere_xc           = sphere_xc0 + time * vel_x;
     sphere_yc           = sphere_yc0 + time * vel_y;
     sphere_zc           = sphere_zc0 + time * vel_z;
     sphere_vxc          = vel_x;
     sphere_vyc          = vel_y;
     sphere_vzc          = vel_z;
```

```
#
#     YAW and YAW-RATE
#
        sphere_psi          = time * psi_dot;
        sphere_psi_dot   = psi_dot;
#
#     PITCH and PITCH-RATE
#
        sphere_theta      = time * theta_dot;
         sphere_theta_dot = theta_dot;
#
#     ROLL and ROLL-RATE
#
       sphere_phi          = time * phi_dot;
       sphere_phi_dot   = phi_dot;
    };

]
```

STORESIM/TETMESH Training Session

General

STORESIM/TETMESH Training Session

Agenda

Session 1:

- Introduction – Project Goals, Status & Future Plans
- Software Components
  - STORESIM
    - – TETMESH
    - – STOREFLOW
- TETMESH Introduction
- STORESIM Demonstration
- STORESIM Introduction

Session 2:

- TETMESH
  - – Example 1 – Concentric Spheres
  - – Example 2 – Wing in a wind tunnel

STORESIM/TETMESH Training Session

Agenda (contd.)

- Session 3:
  - Store Separation
    - Mesh Movement
    - Algorithmic Issues
  - STOREFLOW
    - Features
    - 6DOF Dynamics
  - Mesh Movement Demo
    - Forced Trajectory
    - Computed Trajectory
- Session 4:
  - Modeling Complex Geometry (IGES input, zero–thickness surfaces, extruded prismatic surface elements, examples)
  - Simulation Run
  - Open Discussion

STORESIM/TETMESH Training Session

Introduction – Project Goals, Status & Future Plans

Goals: Develop a capability to simulate multi–body CFD using state–of–the–art methods:

- unstructured grids
- vectorized flow solvers
- geometry–based grid generation
- 6DOF solvers

Research into each of the above

- IMPLEX
- Delaunay schemes in 3D
- Mesh movement/restructuring
- real gas
- multi–element types

STORESIM/TETMESH Training Session

Introduction – Project Goals, Status & Future Plans

Status:

Project nearing the end
Initial SOW features verified (with a few small exceptions)
All SOW features independtly verified – interfacing continues

Flow Solver:

- Multiple element types (hex, tet, prism, quad, tri)
- inviscid, viscous
- algebraic turbulence model
- real gas (equilibrium chemistry)
- explicit, implicit/explicit

Grid Generator:

- surface triangulation
- tetrahedralization using Delaunay
- face enforcement
- geometry–based

STORESIM/TETMESH Training Session

Introduction – Project Goals, Status & Future Plans

Status:

Moving Bodies:

- grid movement
- grid restructuring

6DOF Solver:

- Runge–kutta 1st, 2nd, 4th order integration
- integrated with mesh mover, flow solver
- consistent force & moment calculations

Visualization:

- iso–surfaces, slicing planes, velocity vectors
- grid, wireframe
- hardware graphics

COMCO

STORESIM/TETMESH Training Session

Introduction – Project Goals, Status & Future Plans

Future Plans:

Grid Generator:
- continue expansion of geometry capabilities
- anisotropic Delaunay
- better face enforcement

Flow Solver:
- integration of viscous, real–gas, turbulence capability into STORESIM
- integration of IMPLEX into STORESIM
- efficiency/tuning

6DOF:
- integration of powered flight

Visualization:
- better time–dependent displays

COMCO

TETMESH capabilities:

- input geometry definition in standard formats:
  - o IGES trimmed surface format
  - o TETMESH E3S trimmed surface format
  - o Network (or 'panel–code') format (converted to E3S)
- surface triangulation in both parameter and physical space
- Delaunay tetrahedralization of domain interior (with boundary face enforcement)
- boundary condition definition by input surfaces
- zero–thickness surfaces
- internal boundaries
- Motif–based interactive GUI
- grid visualization (with hardware graphics support)

STORESIM capabilities:

- flow solver:
  - o inviscid
  - o viscous
  - o algebraic turbulence model
  - o real–gas model
  - o explicit, and implicit/explicit solvers
  - o multiple element types (TET, PRISM, HEX, QUAD, TRI)
  - o stand–alone or integrated with STORESIM/TETMESH
  - o fully vectorized
- six–degree–of–freedom (6–DOF) solver
  - o 4th–order Runge–Kutta ODE solver
  - o tightly coupled with flow solver for consistent forces & moments
  - o free– or powered–flight specification
  - o component specification through TETMESH
- grid mover
  - o grid motion in response to boundary movement
  - o grid–flow analogy
  - o automatic grid restructuring to avoid grid tangling

- visualization
    - static
        - planar slice
        - iso–surfaces
        - velocity vectors
        - transparency through graphics hardware
    - dynamic
        - user–selection of time value to view solution
- Motif–based interactive GUI
- grid visualization (with hardware graphics support)

TETMESH

- o Hiearchical Approach
  - – Discretize Edges
  - – Enforce Edges
  - – Discretize Surfaces
  - – Enforce Surfaces
  - – Tetrahedralize Boundary nodes
  - – Enforce faces if needed:
    - > enforce face edges first
    - > enforce faces
  - – Insert Interior nodes and tetrahedralize (Bowyer scheme)

TETMESH

- o Edge Discretization
  - – Manual editing:
    - > number of nodes
    - > average spacing
    - > clustering:
      - – uniform
      - – input (based on input distribution of edge knots)
      - – Sine clustering: $x + \alpha \sin(\beta x)$
      - – Hyperbolic tangent: can specify spacing at each end of the edge independently
  - – Automatic discretization
    - – by 1–D curvature
    - – by average of attached surface curvatures (preferred due to isotropic Delaunay scheme – ensures good quality triangles in regions of high surface curvature)
- o Edge Enforcement
  - – Delaunay criterion: edge segment will appear in final tetrahedral grid if its minimal circumsphere contains no other nodes
  - – Only refine if nodes are in its circumsphere on the 'interior' side of the domain based on local surface normal

TETMESH

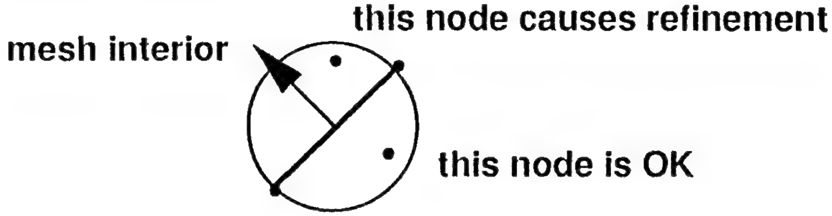
- o Discretize Surfaces
  - – Triangulate in UV–space (parameter space) using Bowyer scheme with nodal insertion at circumcircle centers
    - > AVL–tree used to dynamically sort triangles based on refinement criterion
    - > refinement criteria are user–selectable options
  - – Options:
    1. only based on linear interpolation of edge spacing into the interior
    2. user–specifed 'average' spacing plus extra passes:
       - 2a. Flatness measure (distance of triangle from true surface)
       - 2b. Curvature measure (from support–surface)
  - – Spacing:
    - > based on UV–space only
    - > based on scaled UV–space: circumcircle radius in UV–space is scaled by local ratio of xyz–space to UV–space metric
      - – gives better 3D surface triangulation

TETMESH

- o Discretize Surfaces (contd.)
  - – Triangulate in xyz–space, using the 3D triangle face circumsphere centers for nodal insertion coupled with 3D diagonal swapping.
    - > gives best 3D surface triangulation

TETMESH

- o Enforce Surfaces
  - – Diagonal swapping based on 3D minimal circumspheres of each triangle face
  - – Additional Option: recursively refine triangles whose minimal circumspheres contain other nodes (in the 'interior' normal direction)



- o Tetrahedralize Boundary nodes
- o Enforce faces if needed:
  - – enforce face edges first by inserting nodes, local tetrahedralization of cavity
  - – enforce faces if needed, again by node insertion
- o Insert Interior nodes and tetrahedralize (Bowyer scheme)
  - – spacing based on surface spacing linearly interpolated into the interior

TETMESH

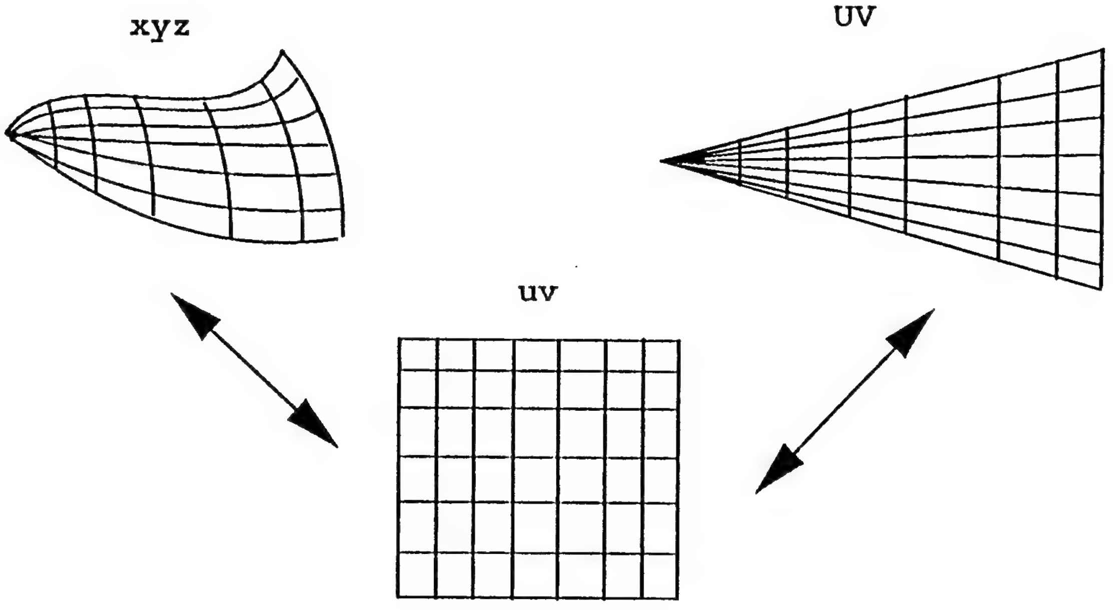
o Geometry

- Trimmed Surfaces
    - support surface: wireframe or NURBS
    - trimming curves: point–defined line or B–Spline
        - fully general topology allowed
    - surface derivatives (first– and second–order by finite–difference)

- Non–uniform Parameter Space Option
    - UV–space matches topology of xyz–surface (i.e. if one edge is collapsed, UV–space has collapsed edge)
    - UV–space wireframe spacing matches xyz–space by iteration/smoothing scheme
    - UV <––> xyz mapping:

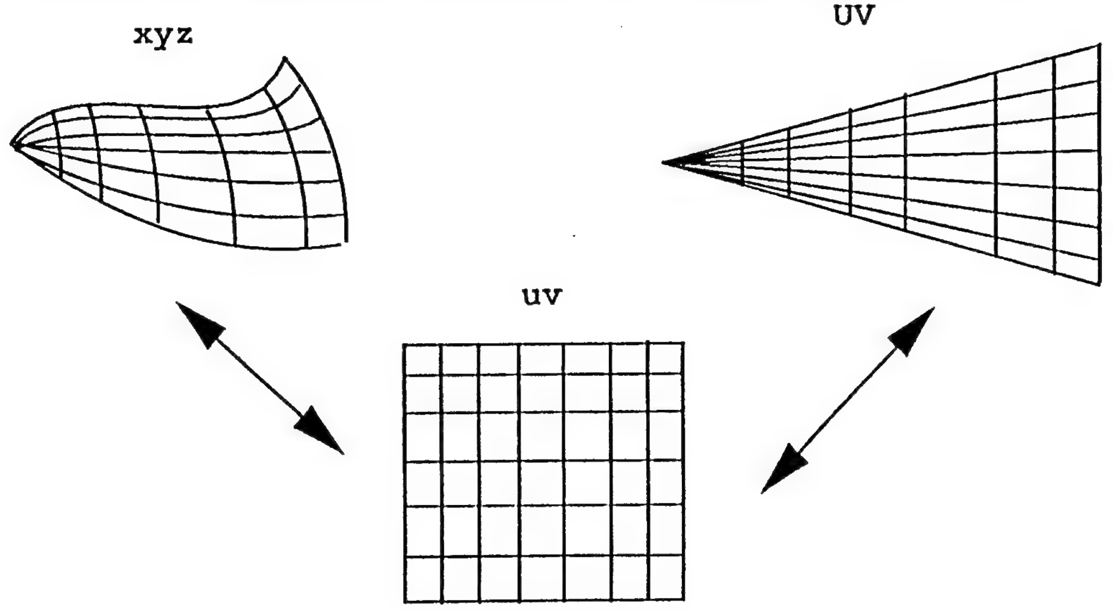
TETMESH

- o Geometry
  - – Non–uniform Parameter Space
    - > UV <––> xyz mapping:
      - – canonical parameter space: uv–space, [0,1] x [0,1]
      - – mappings from UV to xyz and vice versa go through uv–space:



TETMESH

- o Geometry
  - – Efficiency:
    - > UV– and uv–coordinates for each wireframe point and each triangulation node are stored
    - > Quad–tree used to enable fast spatial search for mappings
      - – localize cell containing xyz (or UV if mapping UV ––> xyz)
      - – perform Newton iteration on mapping function



TETMESH

o Geometry
  - Input:
    - 'Net' format: standard 'panel', 'patch', 'wireframe' format:
      - set of (m x n) xyz coordinates
      - does not allow for trimmed–surfaces (e.g. holes)
      - match of two networks across a common edge: 3 points equal
    - 'E3S' format (internal): net format is converted to this format
      - three components:
        > SUPPORT–SURFACE
          - wireframe (m x n) defines shape, OR
          - NURBS surface (WP–FIMM IGES Entity 128)
        > EDGE
          - set of xyz points defining trimming curve
          - B–spline curves (WP–FIMM IGES Entity 126):
        > TRIMMED–SURFACE
          - set of 'boundaries'; boundary is a set of EDGE's
          - IGES 143/141 for trimmed surface specification

TETMESH

o Geometry (cont'd)

> COMPONENT: set of TRIMMED–SURFACES forming a closed body
>
> + component name attaches to a 'HOC program' for trajectory specification
> + Possible trajectory computation flags:
>
>   NO–MOTION
>   TRAJECTORY–SPECIFIED
>   SIXDOF–SOLVER–CALCULATE

– HOC program: C–like symbolic interpreted language

> mass–properties specification: mass, I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}

> motion specification as function of time: xyz of center–of–mass, linear and rotational velocity of body

> thrust forces and moment specification as function of time

TETMESH

- o Geometry (contd.)
  - – 'E3S' format (contd.)
    - > Boundary Conditions:
      - – name, type, data–pack, trimmed–surface list to apply BC to

STORESIM/TETMESH Training Session

Classroom Examples

Test Case No. 1:

Name: Concentric Spheres

Description:

This case consists of a spherical rigid body of unit diameter embedded in a spherical flow domain. The diameter of the outer sphere is 10 units. The purpose of this test case is twofold:

1. Provide a basic exercise in mesh generation and flow boundary condition specification using TETMESH.

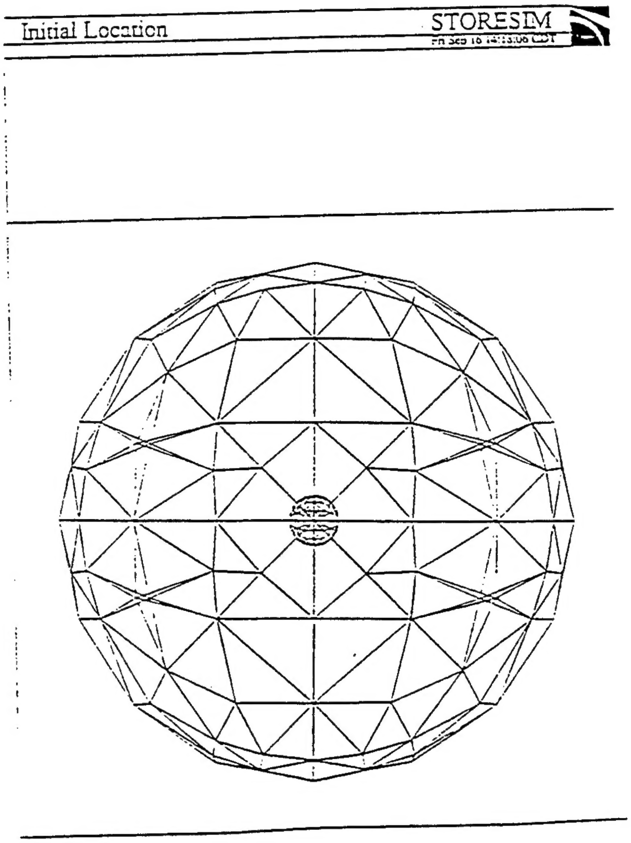
2. Create an integrated rigid–body dynamics and fluid flow simulation example.

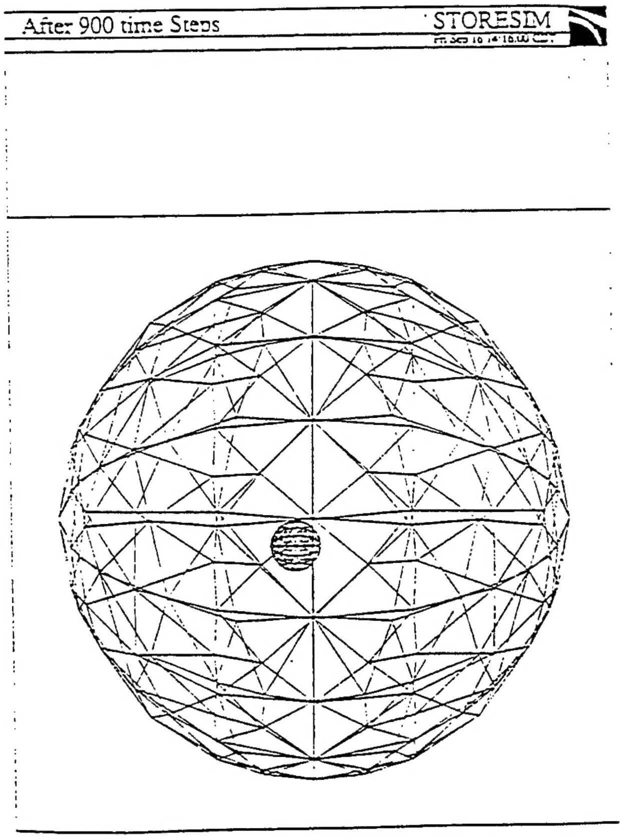
Features: General, Multiple bodies, Coupled fluid flow, rigid body motion, and mesh restructuring.

Input File: ./sphere/sphere–patch.net

Mesh Generation Procedure Outline:

a. Convert the network file to E2S format. (Resulting file: net2e2s)
b. Convert net2e2s file to E3S format.
c. Impose boundary conditions. (Edit the E3S file).
d. Run TETMESH to generate the mesh. (See Flowchart).





Test Case No. 2:

Name: Flow over a Delta wing

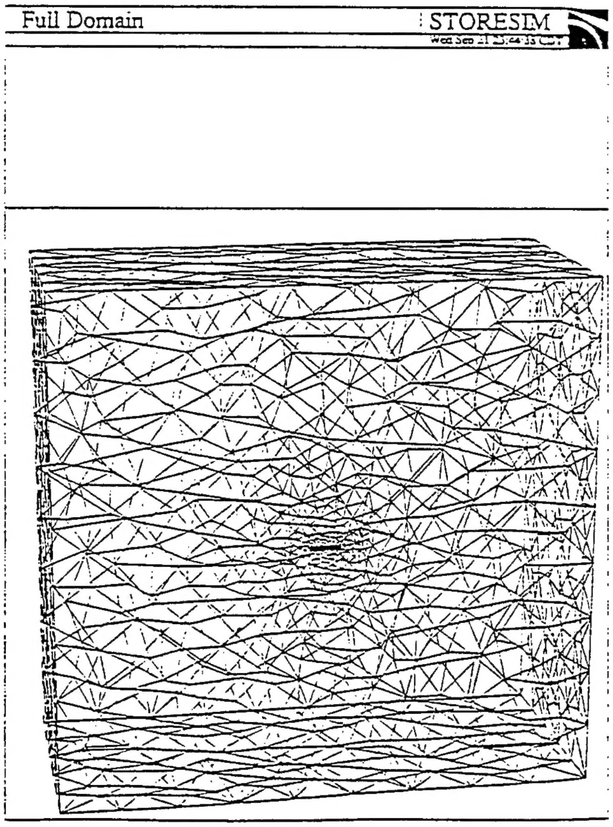
Description: This case consists of a Delta wing attached to a wall at the root. The purpose of this test case is to mesh the flow domain using TETMESH and analyze the flow field using STOREFLOW. The incident flow in this case is at Mach 2 and is at angle of 10 degrees.

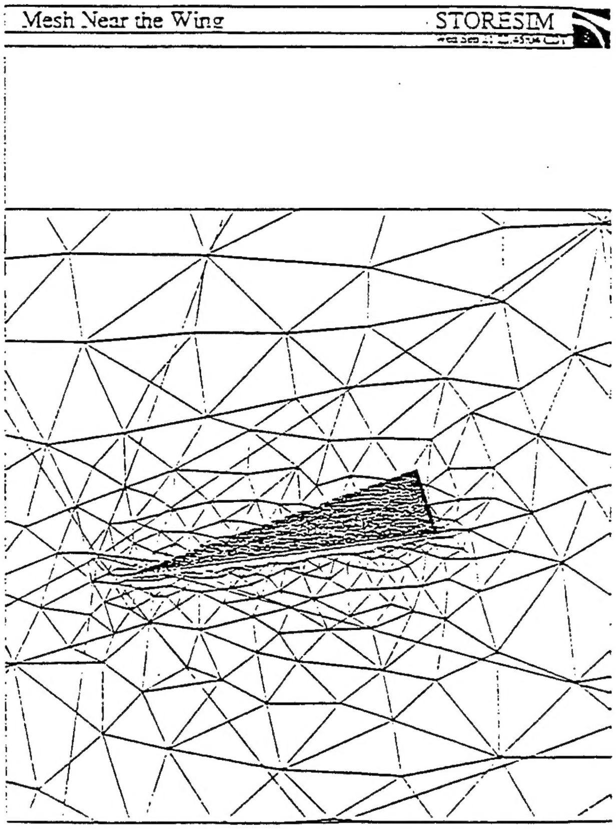
Features: General, trimmed patches, flow solver parameter specification, typical flow simulation.

Input File: ./Delta_wing/delta.e3s

Simulation Procedure Outline:

a. Familiarize with the boundary conditions format. (Read the E3S file).
b. Run TETMESH to generate the mesh. (See Flowchart).
c. Initialize the flow field to Inflow conditions.
d. Run the flow solver. (Refer to TETMESH widget, Fire Storeflow button)
e. Postprocess the existing flow solution. (Readin the save files)





Test Case No. 3:

Name: Cube with an extended zero thickness surface inside a cube

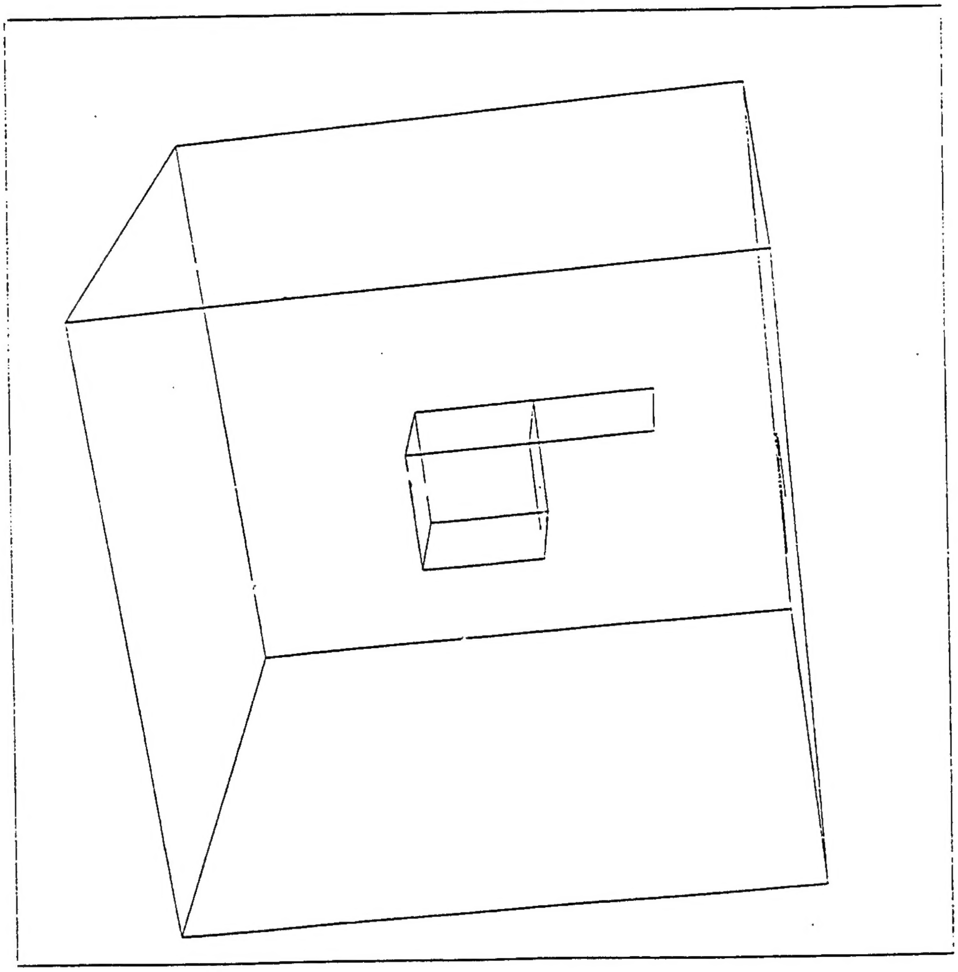
Description: In this test case a smaller cube is placed inside a larger cubical flow domain. The inner cube has an extended flap which is a **zero thickness** surface. The main purpose of this exercise is to demonstrate the capability of TETMESH to recognize and appropriatly treat such surfaces.

Features: General, handling zero thickness surfaces.

Input File: ./zero_thickness/cube–0–in–box.e3s

Mesh Generation Procedure Outline:

a. Familiarize with the specification of zero thickness surfaces.
b. Plot normals and note the double sidedness of the zero thickness surface.
c. Run TETMESH to generate the mesh. (See Flowchart).



Test Case No. 4:

Name: Finned store

Description: This test case has a relatively complex geometry. It comprises of several multiply connected surfaces. It provides a good example for familiarizing oneself with several basic features of STORESIM/TETMESH.

Features: General, Multiple bodies, Generation of trimmed patches, coupled flow solver, with rigid body motion and mesh restructuring.

Input File: ./finned_store/wpfsurf.net

Mesh Generation Procedure Outline:

a. Convert the network file to E2S format. (Resulting file: net2e2s)
b. Convert net2e2s file to E3S format.
c. Impose boundary conditions. (Edit the E3S file).
d. Run TETMESH to generate the mesh. (See Flowchart).
e. Postprocess the existing flow solution. (Readin the save files)